Multi Agent Reinforcement Learning with Large Language Models for Safe Path Planning

Graves Reid, Allen Chase, Krishnan Shawn, Zhao Eric, Ali Jonathan
{rgraves, mcallen2, shawnakk, ericzhao, jhali}@andrew.cmu.edu

ABSTRACT

Recent advancements in hierarchical multi-agent reinforcement learning (MARL) frameworks have enabled quadrupedal robots to perform complex tasks such as long-horizon, obstacle-aware object pushing [1]. However, these frameworks lack the ability to dynamically adapt to sensitive and unpredictable environments, where interactions with obstacles of varying severity levels pose significant safety and efficiency challenges.

In this work, we extend a state-of-the-art MARL framework by integrating large language models (LLMs) as high-level controllers to introduce context-aware planning and execution capabilities. Using GPT-4, the system classifies environmental objects by severity and dynamically adjusts penalties for approaching sensitive obstacles [2]–[4]. These penalties guide mid-level policies and influence the RRT-based high-level planning, ensuring safe and efficient path generation while maintaining task performance. Unlike traditional frameworks, our approach introduces semantic reasoning to robotic decision-making, creating a system capable of safely interacting with unstructured, dynamic environments [5]. Preliminary results demonstrate improved obstacle avoidance for sensitive hazards, paving the way for further exploration of LLMs as context-aware components in robotic systems.

I. INTRODUCTION

Recent advancements in quadrupedal robots and hierarchical multi-agent reinforcement learning (MARL) frameworks have enabled robots to achieve complex tasks such as long-horizon, obstacle-aware object pushing in multi-agent settings [1]. However, existing frameworks often lack the capability to dynamically adapt to environments with sensitive obstacles, such as humans or objects of varying importance, limiting their operational safety and efficiency in real-world scenarios. To address this gap, we propose integrating large language models (LLMs) into the high-level controller of a hierarchical MARL framework to enhance context-aware planning and safety during task execution.

Our approach focuses on improving operational efficiency and safety by incorporating the contextual dangers associated with interacting in sensitive environments. For example, sensitive obstacles like humans are assigned high severity levels, prompting greater operating distances. The LLM enhances subgoal generation indirectly by leveraging severity levels to assign penalties for approaching obstacles, which are incorporated into the reward structure of the mid-level policy and used to guide the RRT planner [6]. This creates a system capable of prioritizing safety and efficiency, enabling trustworthy decision-making in critical applications while fostering user confidence and operational reliability.

The LLM, GPT-4, is utilized for interpreting the prompts describing objects in the environment to classify obstacles as low, medium or high severity. These classifications dynamically update the penalties associated with approaching different obstacles, enabling robots to navigate around them more effectively. While the LLM does not directly generate subgoals, its semantic understanding significantly refines obstacle-aware planning through the integration of classified obstacles, particularly in unstructured environments where sensitive obstacles may move unpredictably. This integration adds a layer of interpretability and adaptability previously absent in MARL frameworks.

This proof-of-concept work introduces a novel application of LLMs as controllers in multi-agent robotic systems, focusing on dynamic penalty scaling rather than traditional motion planning or direct subgoal generation. To our knowledge, this is the first attempt to augment hierarchical MARL with LLMs for obstacle composition-based path planning. Despite limitations, such as the non-deterministic nature of LLM agents, this approach lays the groundwork for future studies to enhance robot safety and decision-making capabilities. By integrating semantic reasoning into hierarchical controllers, we aim to extend AI safety and contextual intelligence to multi-agent robotic systems, with applications ranging from industrial automation to human-robot interaction in sensitive environments.

II. RELATED WORKS

A. Isaac Gym

Isaac Gym [7] is a GPU-accelerated physics simulation platform designed for robotics learning and rein-

forcement learning (RL). It integrates physics simulation and neural network training entirely on the GPU, bypassing CPU bottlenecks, and enabling training speeds orders of magnitude faster than traditional CPU-based simulators. By maintaining all computations on the GPU, Isaac Gym supports tens of thousands of environments running in parallel on a single GPU, allowing researchers to conduct large-scale experiments locally [7]. The platform leverages NVIDIA PhysX for high-performance physics simulation and offers a PyTorch-compatible tensor API to efficiently integrate observations and actions with RL frameworks. This end-to-end GPU pipeline enables contact-rich robotic tasks, such as dexterous manipulation and humanoid locomotion, to be trained rapidly. These types of tasks are trained in the IsaacGym environment using walktheseways [8] for sim-to-real RL tasks such as those addressed in this paper.

B. Loco-Manipulation for Legged Robots

Recent advances in quadrupedal robots have focused on improving locomotion and manipulation capabilities [9]–[14]. Research in prehensile locomanipulation often integrates grippers or robotic arms to expand operational capabilities, while nonprehensile approaches use legs or the robot's head for manipulation tasks. While these methods enhance dexterity and payload handling, they often neglect complexities in multi-agent scenarios with large, dynamic objects.

Efforts to combine locomotion and manipulation with hierarchical reinforcement learning frameworks based on optimization (HRL) have shown promise [2]–[4]. Murooka et al. introduced a planning framework for humanoids to push large objects [15], while Rigo et al. developed hierarchical control to optimize robot contact strategies [3]. These contributions demonstrate the potential of structured control frameworks but lack the adaptability required for dynamic environments.

C. Multi-Agent Collaborative Manipulation

Multi-agent systems have demonstrated success in collaborative manipulation, leveraging decentralized control and reinforcement learning. Nachum et al. proposed a two-level MARL framework for object-centric navigation, while Xiong et al. explored MARL in competitive tasks. However, these methods often neglect safety-critical considerations, such as obstacle sensitivity and dynamic penalties, which are crucial in real-world applications.

Hierarchical frameworks for quadrupedal manipulation, such as that of Hong et al., have made significant strides in achieving obstacle-aware, long-horizon object pushing. Their three-level architecture, integrating RRT-based planning and decentralized policies, effectively handles tasks like obstacle avoidance and long-distance object pushing. However, it lacks context awareness when interacting with sensitive obstacles, such as humans, which limits its safety applications.

D. Hierarchical Reinforcement Learning

HRL methods have been used to address long-horizon tasks, with high-level policies generating sub-goals and low-level controllers managing physical actions. Such frameworks are particularly effective in multi-agent settings, where centralized or decentralized policies coordinate agent behaviors. Hong et al. applied a hierarchical MARL framework with high success rates, yet its ability to adapt dynamically to unstructured environments remains constrained [8].

E. Multi-Agent Collaborative Manipulation for Long-Horizon Quadrupedal Pushing

These prior works culminate to the recently published Multi-Agent Collaborative Manipulation for Long-Horizon Quadrupedal Pushing (MA Push) project, which reflects the current state of multi-agent prehensile object manipulation. MA Push leveraged a three tier hierarchy that enables effective large-object manipulation across different objectives and examples. At the top, an RRT planner generates a geometrically feasible trajectory that neglects factors such as robot push capability and other agents in the environment. The high-level adaptive policy then uses this trajectory as a reference to assign a sub-goal to the target object, ingesting a universal state vector to assign commands to a low-level policy responsible for assigning joint torques and facilitating locomotion. The RRT planner is executed only once at the start of each episode given its computational overhead.

MA Push introduced a robust hierarchical MARL framework for long-horizon, obstacle-aware object pushing, showcasing the potential of adaptive subgoal generation using an RRT-based planner and reward-driven decentralized policies. However, its design assumes static penalties for obstacles, limiting adaptability in environments where obstacle sensitivity varies [1]. For example, interactions with dynamic and sensitive obstacles like humans or fragile objects require nuanced hazard modeling and context-sensitive penalties, which MA Push lacks. Our work addresses this limitation by augmenting MA Push with context-aware navigation and hazard adaptation through large language models (LLMs).

III. METHODOLOGY

A. LLMs for Object Severity Classification

To minimize reliance on predefined contextual awareness of an environment, we propose integrating LLMs

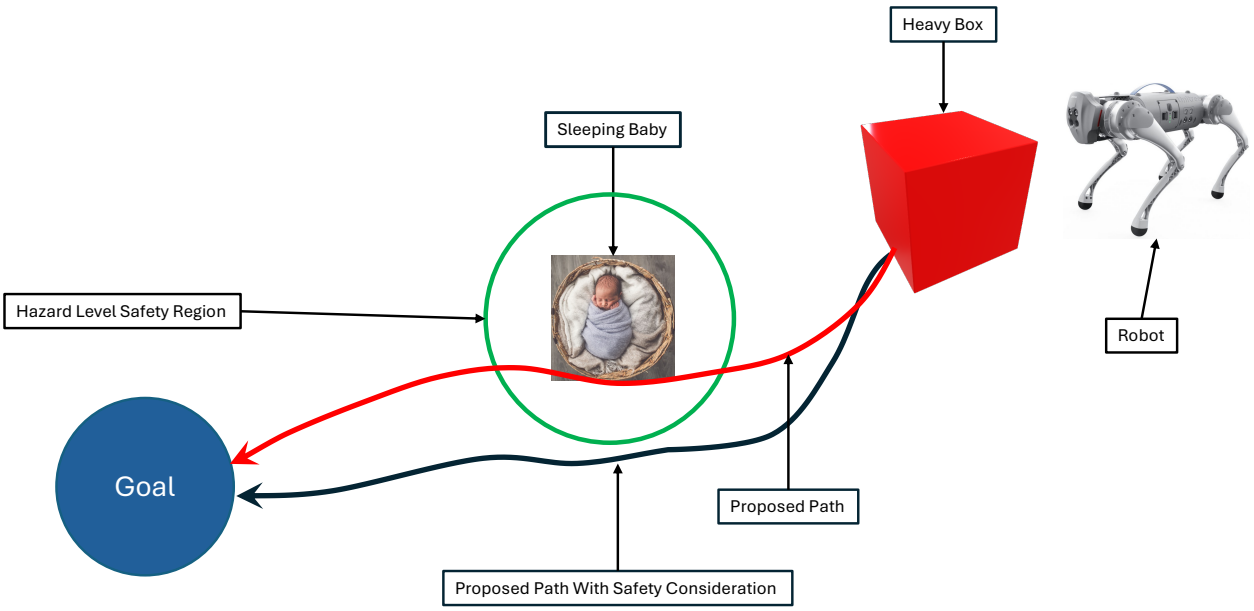

Fig. 1: Proposed target path augmentation, taking into account obstacle context

into the pre-planning stage. This approach enables dynamic environmental recognition without requiring a large dataset containing context-specific information or the computational overhead of processing it. By connecting to pre-existing APIs, robots can access up-to-date, contextually relevant information in real time, significantly improving their path-planning capabilities. This not only reduces the dependency on extensive data collection and preparation, but also enhances robots' ability to adapt quickly to dynamic and unstructured environments, ensuring efficient and safe navigation.

In order to apply this concept into practice, we utilized the Open AI GPT-4o-mini, and GPT-3.5 model API's in order to retrieve severity metrics of objects that would be located within an environment. Further analysis to evaluate the metrics associated with various model's effects on the speed and reliability of using LLM is beneficial.

We determined that due to modern advances in computer vision, we would be able to determine what objects would be present in an environment [5] so we can reduce the informational dependency on LLM models with this advancement in research. Given we have a list of objects in an environment, we can then make an API call to an LLM and allow it to label it with a "severity." In this context the use of the word "severity" refer's to a tri-label classification as to how far the robot should maintain distance from a object which is also implemented within the reinforcement learning path planning control. We sent a customized prompt as the API call to the LLM and then obtained a list with severity of the various objects within the environment and created a JSON file containing this information to be passed to the High-Level RL path planner. We also incorporated a failsafe default classification in the event that an LLM call returns an unusable response, which returns all values defaulted to medium.

One of the most integral aspects of using LLMs for this purpose is to craft prompts that are capable of consistently returning the requested information. This is known as zero-shot prompting: [16] In our approach, we augmented the principles of zero-shot prompting with elements of few-shot prompting, designing prompts that achieved a 94.1% success rate in retrieving the desired severities on the first API request. This hybrid strategy maximized reliability and minimized computational overhead. This hybrid strategy maximized reliability and minimized the number of API calls we were making, which would lower the cost of running these programs on scale.

LLM integration enables dynamic hazard penalty adjustment and path optimization for safer navigation. By bridging semantic reasoning with RL-based control, our method introduces an additional layer of safety and adaptability to hierarchical MARL frameworks.

B. Reward Function Augmentation

To enable robots to perform long-range collaborative tasks while considering context-sensitive obstacle penalties, we propose an enhancement to the existing MA Push framework that provides flexible sensitivity to varying objects in the environment. The enhanced framework consists of three hierarchical controllers, described as follows: [1].

High-Level Controller: The high-level controller integrates the RRT planner for global trajectory planning as defined in MA Push. We augment this controller with a GPT-4o-mini-based module that classifies obstacles into severity levels (low, medium, high) based on environmental context and object descriptions. These classifications dynamically scale the risk penalties, which directly influence both the subgoal generation process and the reward structure for safe navigation.

Mid-Level Controller: The mid-level decentralized policy uses the adjusted rewards of the high-level controller to guide agents toward sub-goals. These subgoals are influenced by hazard penalties derived from obstacle sensitivity, ensuring that agents prioritize safe navigation around sensitive obstacles while maintaining task performance.

Low-Level Controller: The low-level controller executes motor commands derived from mid-level velocity targets, as described in MA Push. This layer remains unchanged from the original framework and focuses on robust locomotion control for individual robots.

C. Reward Design

1) Mid-Level Reward: The mid-level reward function is the same implementation used in [1]. The reward function $r^m = r^m_{task} + r^m_{penalty} + r^m_{heuristic}$, is a sum of the reward for a mid-level task, the penalty, and a heuristic reward function. The task reward r^m_{task} incentivizes actions that move the object toward the target point. The penalty term $r^m_{penalty}$ penalizes agents for close proximity, robot falls, and timeouts. The mid-level heuristic reward $r^m_{heuristic}$ is the sum of a mid-level approach reward, a velocity reward, and an occlusion-based reward [.....]. This heuristic reward is represented as $r^m_{heuristic} = r^m_{approach} + r^m_{vel} + r^m_{OCB}$. The midlevel approach reward, $r^m_{approach}$, provides encouragement for agents to approach the object, while the implementation of the velocity reward r^m_{vel} introduces a predefined velocity threshold which, when exceeded, rewards agents and promotes greater diversity and stability in pushing actions. The conclusion-based reward r^m_{OCB} is as defined in [1], where it is implemented to lead agents to more optimal contact points in situations where robots' views of subgoals are obstructed. This reward term encourages agents to target occluded surfaces, which can lead to more effective push behavior.

2) High-Level Reward: The high-level reward function is an augmented version of the MA Push high-level reward implementation, defined as $r^h = r^h_{task} + r^h_{penalty}$. It consists of a high-level task reward term and a penalty term. The task reward term r^h_{task} sparsely rewards the achievement of the target goal and has more dense rewards for the minimization of the distance between the sub-goal and the nearest point on the RRT trajectory and for reducing the distance between the target and the object. This framework allows for adherence to the RRT path while providing flexibility for complex push tasks. The high-level penalty term $r^h_{penalty}$ incorporates penalties for various cases, including close proximity to obstacles and heavier penalties for certain exceptions such as agents falling over, colliding, object tilting, and timeouts. In addition to this existing high-level reward framework, we introduce an additional penalty term that penalizes agents based on their proximity to an object given its hazard severity classification determined by the GPT-4o-mini LLM module. The objects have a given hazard radius, which is a scalar value defined by the hazard level of the object and the size of the object itself. Using the Euclidean distance between the agent and the center of an object, we can determine whether the agent has entered an object's hazard zone. This incurs a penalty, which scales with how deep the agent is inside the hazard zone (i.e., how close they are to the object).

Evaluating LLMs for Simple Path Planning in Multi-Agent Systems:

In our work, we assessed the efficacy of Large Language Models (LLMs), specifically OpenAI's GPT-4o-mini, as a tool for path planning in multiagent systems, particularly under conditions requiring obstacle awareness. We leveraged conversational abilities of the LLM to guide two agents through a series of 2D waypoints, where optimal paths are generated between given start and end points, considering step sizes and other constraints.

To evaluate the model, we developed a series of path-planning interactions with the LLM, where the agents were tasked with navigating from a starting point to a target location in a two-dimensional space. The LLM was asked for specific coordinates for each agent within a proposed 5.0 m x 6.0 m grid and was tasked with providing the most efficient path between those points. The agents' paths were calculated based on step sizes, which the LLM generated, and plotted using matplotlib for visual verification. The core of the experiment centered around the assistant's ability to generate realistic paths and navigate obstacles with safety in mind.The LLM's outputs were then compared to the "actual" path calculated using traditional Euclidean distance methods with specified step sizes, allowing us to evaluate how close the agent's generated path was to optimal. The results of this comparison were visualized through graphs showing both LLM-generated and calculated linear paths.

The system initiates with a user input for start and end points, followed by the LLM interaction to generate coordinates. The assistant's responses are parsed and visualized using a plotting tool, comparing the generated path to the optimal path calculated based on step size. The results were stored in a structured

JSON file, allowing easy tracking of both user inputs and assistant responses, including path coordinates. This method represents how well LLMs can contribute to the decision-making and coordination of multi-agent systems, where agents need to autonomously generate paths while also considering dynamic and potentially safety-critical obstacles in real-world environments.

Our results showed that, while the LLM successfully generated paths between the designated points, deviations were observed when compared to the calculated optimal path. For instance, when navigating from points (0, 0) to (2, 3), our generated linear paths had a maximum y-error deviation of 5 percent, However, from (2, 3) to (7, 8), we witnessed a 14 percent maximum x-error deviation. This discrepancy highlights the challenges of relying on a language model for precise numerical tasks like path planning. However, in environments where obstacle awareness and real-time adaptability are paramount, the LLM demonstrated potential for supporting decision-making by enabling agents to dynamically classify obstacles and adjust their behavior accordingly.For instance, the LLM's semantic understanding can aid in interpreting the presence of obstacles (e.g., humans or moving objects) in the environment, as demonstrated in the prior work of hierarchical reinforcement learning frameworks. The integration of LLMs for obstacle classification can be instrumental in multi-agent settings, where safety concerns such as collision avoidance are crucial. Future work could further explore integrating LLMs with reinforcement learning-based agents, enabling them to generate subgoals based on obstacle severity and the dynamic nature of the environment, as discussed in the context of multi-agent reinforcement learning (MARL) frameworks.

IV. Experiments

A. LLM Setup

In setting up the LLM there where 2 main decisions of how we could use the LLM for severity classification. The first was determining where or not to use a nominal or categorical classification. The second decision was in how we evaluated our performance of the LLM's classification. We created multiple appraches and abodpting a final approach after evaluating multiple methods, including a numeric scale from 1 to 10 and a mapped categorization system where numeric values were grouped into fixed ranges: High (10–7), Medium (6–4), and Low (3–1).

B. LLM Results and Analysis

We assessed LLM classifications by comparing predictions to a set of user-defined ground truth labels. We used a list of semi-randomly generated objects and assigned a ground truth severity listed in the appendix.

We implemented a nominal classification using the prompt: "Only give me a single natural number as your response. On a scale of 1–10 (10 being severe), how dangerous would it be for a free-moving robot to collide with a baby? Be conservative with your analysis!"

In this zero-shot promoting, we did not included any examples are relied sole on the LLM's pre-trained model. The word "baby" was initialized as a variable and could be replaced to represent any object. This method resulted in the LLM achieving an accuracy of 6.45%. We realized that, due to the poor performance, nominal categorization is not a strong point of OpenAI's models. However, more sophisticated methods of evaluation must be employed to produce more definitive results. In this case, we have simply observed that it is a poor choice. (Show the Nominal Single User Classification Graph)

The next method we implemented was a tri-label classification system of ("High," "Medium," "Low"). Our new prompt was: "Please respond with a single word: high, medium, or low. How dangerous would it be for a free-moving robot to collide with a (OBJECT)? Typically, classify living, fragile, or dangerous objects as 'high,' while most other objects fall under 'medium' or 'low.' Be conservative in your analysis." In addition to switching to a tri-label classification we also improved the prompt with two features typically used in few shot prompting. We included a preferred for single word responses, gave examples of types of objects and what their classifications should be, and additionally incorporating a conservative filter in order to give a humanized perspective on how the LLM should be giving safe definitions.

In this new approach we also altered the method of testing by having two accuracies, one for a direct match with the ground truth label, as well as one with 1/2 credit given for over classifying (ie if the LLM returned high but our ground truth was medium) objects. The direct match accuracy, which is the same metric used in the nominal classification was 59.46% which is a significant improvement. And with partial credit accuracy becomes 81.76% which moves us into the relm of this LLM providing useful values. (SHOW THE CATEGORIGAL GRAPH)

We tested a third method that aimed to combine the two approaches, using nominal classification from the LLM, which was subsequently mapped to a categorical classification programmatically.

We used the third and second method to test the resilience of the LLM request by taking each request and looping it 5 times and taking the average of the request. These results indicated that the LLM is fairly consistent in it's response and further showed that the Nominal classification offers less variability.

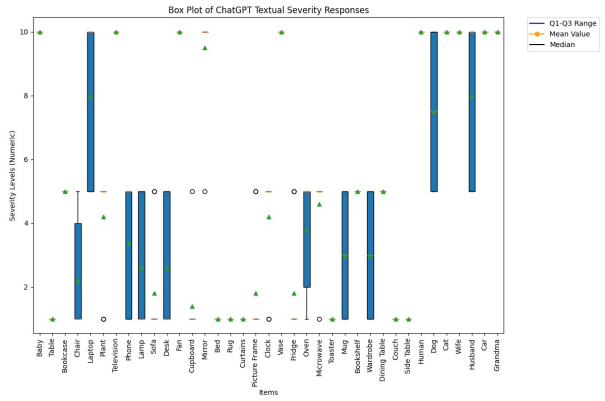

Fig. 2: Box plot of GPT4 classification relative to group's average response (green)

The second major consideration was the labeling of ground truth, as individual biases inevitably influence what a person considers the severity of an object. All methods were initially tested from the programmer's perspective to standardize the process. However, to assess variability and potential improvement with a more collective approach, we compared the accuracy of ChatGPT responses under two perspectives: a one-person perspective and the lab-group consensus perspective (5 people). Under the one-person perspective, ChatGPT achieved an accuracy of 54.05% (exact matches) and 78.38% (partial credit). In contrast, under the group perspective, these metrics increased to 64.86% (exact matches) and 89.19% (partial credit). A two-sample z-test for proportions revealed no statistically significant difference between the two perspectives (p=0.3692, p=0.3692), indicating that while group consensus improves observed accuracy, the variation may not be robust enough to justify altering the ground truth assignment methodology. Future studies could explore alternative methods of reducing bias in severity labeling particularly by using a much larger group sample ground truth.

C. Gym Environment Setup

1) Environment and Task: Similar to MA Push, our simulation environment is built on top of IsaacGym [7]. In it, we consider randomly placed cuboid obstacles (hazards) measuring 1.0m x 1.0m x 1.0m and a target object the 2 quadrupeds need to push. Unitree Go1 robots are utilized in simulation, with a payload capacity of 5kg. The target object to push is a 3kg shaped T-block, which exceeds the robots in size. The positions of the robots, hazards, and T-block are randomly initialized with a target position initialized on the other side of the room. The task is successfully completed if the center of the T-block is within 1m of the target position, and failed if the simulation runs longer than a threshold number of seconds, robots collide with each other or the boundaries, or any object (including the agnets) moved within the set boundary for the hazards.

2) LLM Integration: As a training surrogate for repeated API calls, we propose using uniformly distributed random classifications of an arbitrary obstacle NPC to allow fast and efficient training. This approach eliminates the LLM integration requisite for training. As such, LLMs are not integrated into the training loop and are instead leveraged at the end of training as an input for the trained policy.

3) Training Execution: Training the policies was executed on an NVIDIA 4070 which ran over 500 environments to accumulate over 100M training steps. Snapshots of the model policies (saved at .pt files) are saved every 1M steps for comparison on model progression.

D. Simulation Results and Analysis

Qualitatively, we can observe the results of this in the high level RRT trajectories shown in Figure 4. In the figure, we observe the hazards (black boxes) and their bounding boxes (red boundaires) which scale with the LLM severity classification. Observing the proposed path, notice the diversion from intersecting the bounding box - preserving a safe distance around the hazard relative to its classified severity while achieving the shortest path to the target position.

In Figure 5 we see that the goal resides withing the bounding box, and the planned trajectory stops at the closest point outside of the bounding box - enabling a safe interaction with the proposed hazard.

Comparing these results to a baseline test in 10 (defined as objects with no boundary) we can observe the proposed path closely maneuvers around the hazard to reach the target. As we have established, this may lead to potentially dangerous or unintended behavior from the agents when interacting with sensitive obstacles.

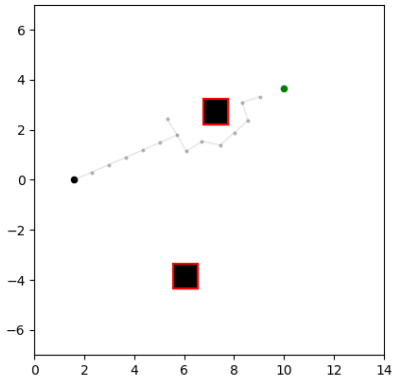

Fig. 3: Baseline RRT path planning with no severity classification

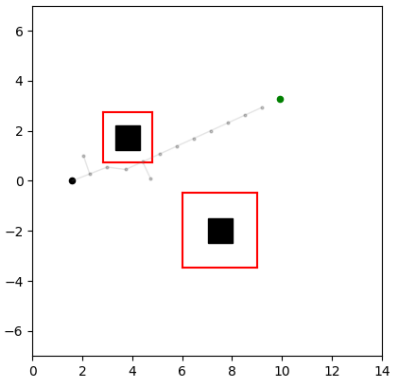

Fig. 4: Avoidance of MEDIUM severity hazard in the environment

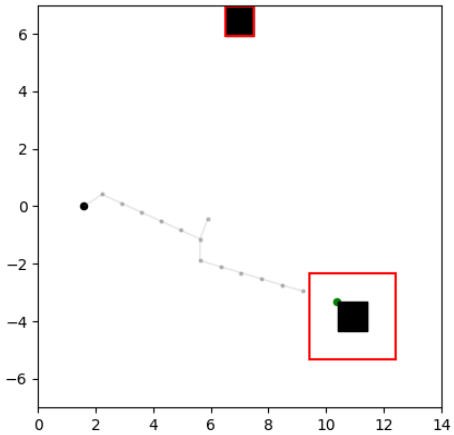

Fig. 5: RRT 100M planned trajectory for infeasible goal on HIGH sensitivity hazard.

Implementing these changes has resulted in improves navigation, with 80% of studies using the updated 100M RRT planner avoiding the tolerance zones established by the LLM. However, our experiments show occasional drift from the target boundary such as in Figure 6 - highlighting a future direction to improve the robustness and safety of our implementation. Even with boundary ingress, our experiments show limited danger towards affected hazards - with agents maintaining a safe distance from vulnerable individuals or obstacles provided the appropriate classification.

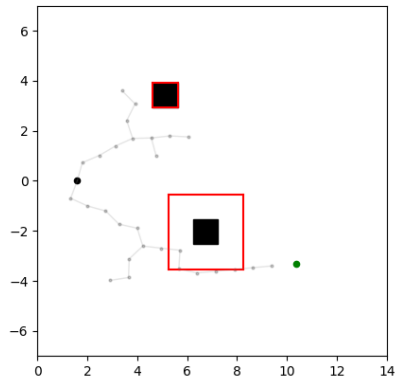

Fig. 6: Planned path with slight failure on bottom HIGH severity hazard

Our analysis of the also highlights further room for improvement in the reward function and training approach. Figure 7 shows that extended training does not correlate with higher task completion after 40M training steps, and the completion rate is highly variable. Despite this, the step reward during training continues to increase in Figure 8, highlighting a potential mismatch between our current reward function and an optimal training configuration to achieve more frequent task completion.

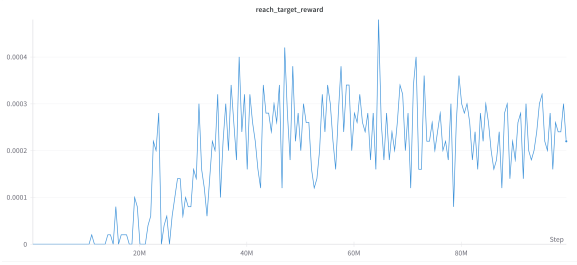

Fig. 7: Completion rate per step during training

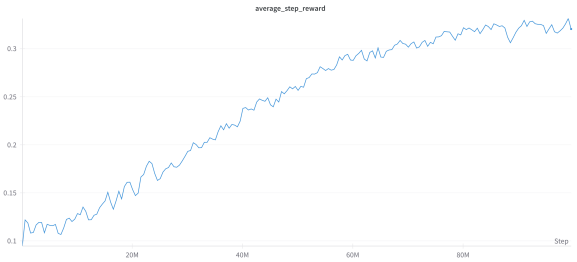

Fig. 8: Step reward during training

V. Conclusion

This work introduces a novel integration of Large Language Models (LLMs) into hierarchical Multi-Agent

Reinforcement Learning (MARL) frameworks to address the critical need for context-aware and safety-oriented path planning. By augmenting the state-of-the-art MA Push framework with GPT-4 as a high-level controller, we demonstrate the potential of LLMs for dynamically classifying obstacles by severity and influencing path planning through adaptive penalty scaling. Our approach preserves task performance while introducing semantic reasoning, enabling robots to navigate unstructured and sensitive environments more effectively. Experimental results validate the feasibility of this integration, showing improvements in safe navigation around hazards and maintaining efficient trajectories. Future work will focus on further refining LLM-based classifications, incorporating real-time decision-making into the training loop, and addressing limitations related to robustness in high-severity scenarios. This proof-of-concept study underscores the promise of combining LLMs with MARL, setting the stage for broader applications in industrial automation, human-robot interaction, and other domains requiring advanced contextual intelligence.

The source code for this project can be accessed here: https://github.com/gravesreid/MAPush

VI. Member Contributions

Reid led the communication and development of this project. His extensive contribution included setting up the original environments for both MQE and MAPush studies, as well as dissecting the respective code bases to make meaningful contributions to by creating original classes, configurations, and experiments which he ran to test changes to the environment and policy.

Chase led the creation of the final report (this document) and assisted in both strategizing and implementing changes to the goal hierarchy for implementing severity classifications in the existing MQE and MAPush environments. Report writing included writing the Abstract, Introduction, Related Works, part of the Methodology, Experiments sections B-D, and the Conclusion. Implementations included experimentation with path planning for both the high and mid-level controllers in MQE, as well as running MAPush experiments with novel reward functions to optimize and cross-validate trained models against Reid's.

Shawn implemented the OpenAI GPT4 integrations and experimental studies, including measuring the team's classification of curated objects and comparing them to a distribution of GPT's classifications. Shawn included these results and explained the methodology for doing so in detail in the report in the Methodology and Experiments sections.

Eric assisted Reid and Chase on MAPush implementation, working to dissect the codebase and provide support for getting the mid-level training running correctly. Eric also contributed significantly to the Methodology section, adding relevant equations and robust description of our approach.

Jon provided a preliminary description of the path planning methodology, creating a framework to build the final section out from. Jon also contributed appropriate citations to the document and assisted the rest of the group throughout the semester wherever appropriate (eg. helping Shawn setup LLM calls, helping Eric to get MAPush running on the lab computer, etc.)

REFERENCES

[1] C. Hong, Y. Feng, Y. Niu, S. Liu, Y. Yang, W. Yu, T. Zhang, J. Tan, and D. Zhao, "Learning multi-agent collaborative manipulation for long-horizon quadrupedal pushing," 2024, arXiv preprint arXiv:2411.07104.

[2] O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar, "Multi-agent manipulation via locomotion using hierarchical sim2real," 2019, arXiv preprint arXiv:1908.05224.

[3] A. Rigo, Y. Chen, S. K. Gupta, and Q. Nguyen, "Contact optimization for non-prehensile loco-manipulation via hierarchical model predictive control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9945–9951.

[4] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 3540–3549.

[5] T. An, J. Lee, M. Bjelonic, F. D. Vincenti, and M. Hutter, "Solving multi-entity robotic problems using permutation invariant neural networks," *CoRR*, 2024.

[6] Z. Xiong, B. Chen, S. Huang, W.-W. Tu, Z. He, and Y. Gao, "Mqe: Unleashing the power of interaction with multi-agent quadruped environment," 2024, arXiv preprint arXiv:2403.16015.

[7] NVIDIA, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021, arXiv preprint arXiv:2108.10470.

[8] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Conference on Robot Learning (CoRL)*, 2022.

[9] S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, and J. Hwangbo, "Learning quadrupedal locomotion on deformable terrain," *Science Robotics*, vol. 8, no. 74, 2023.

[10] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020.

[11] R. Yang, G. Yang, and X. Wang, "Neural volumetric memory for visual locomotion control," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[12] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," 2021, arXiv preprint arXiv:2107.04034.

[13] B. Lindqvist, S. Karlsson, A. Koval, I. Tevetzidis, J. Haluska, C. Kanellakis, A. a. Agha-mohammadi, and G. Nikolakopoulos, "Multimodality robotic systems: Integrated combined legged-aerial mobility for subterranean search-and-rescue," *Robotics and Autonomous Systems*, 2022.

[14] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 11 443–11 450.

[15] M. Murooka, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba, "Whole-body pushing manipulation with contact posture planning of large and heavy object for humanoid robot," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5682–5689.

[16] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," *arXiv*, 2024.

Items	Severity
Baby	10.0
Table	5.0
Bookcase	5.0
Chair	5.0
Laptop	5.0
Plant	10.0
Television	10.0
Phone	1.0
Lamp	10.0
Sofa	1.0
Desk	5.0
Fan	5.0
Cupboard	5.0
Mirror	10.0
Bed	1.0
Rug	1.0
Curtains	1.0
Picture Frame	5.0
Clock	1.0
Vase	10.0
Fridge	10.0
Oven	10.0
Microwave	10.0
Toaster	10.0
Mug	5.0
Bookshelf	5.0
Wardrobe	5.0
Dining Table	5.0
Couch	1.0
Side Table	5.0
Human	10.0
Dog	10.0
Cat	10.0
Wife	10.0
Husband	10.0
Car	10.0
Grandma	10.0

Fig. 9: Single user ground truth table

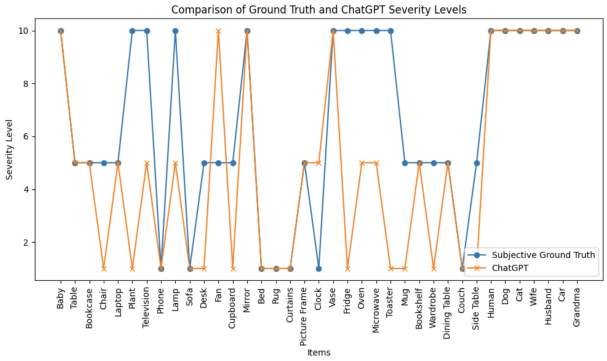

Fig. 10: Ground truth relative to LLM output severity levels preliminary average

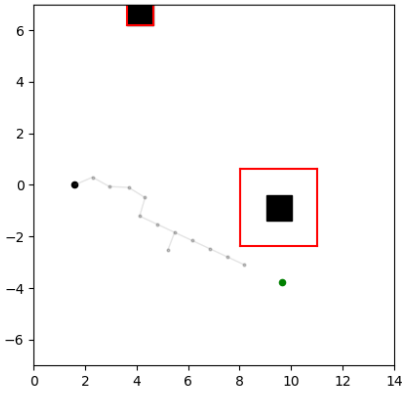

Fig. 11: Path planning for no interference, maintining expectged behavior

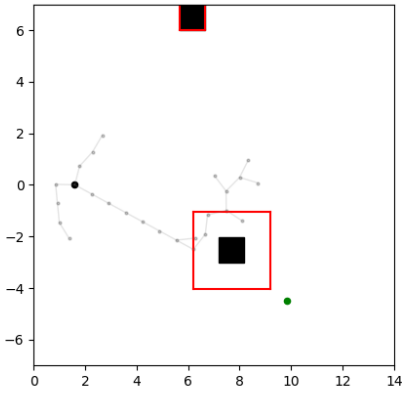

Fig. 12: Path planning for sensitive object directly in optimal path

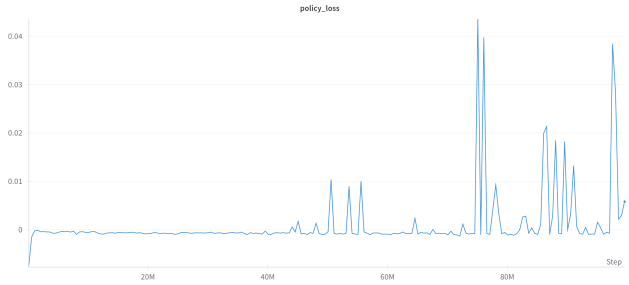

Fig. 13: Policy loss per step

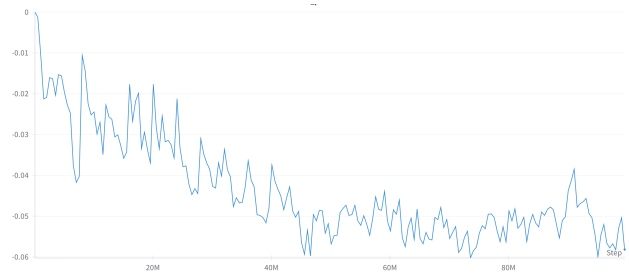

Fig. 14: Hazard punishment per step

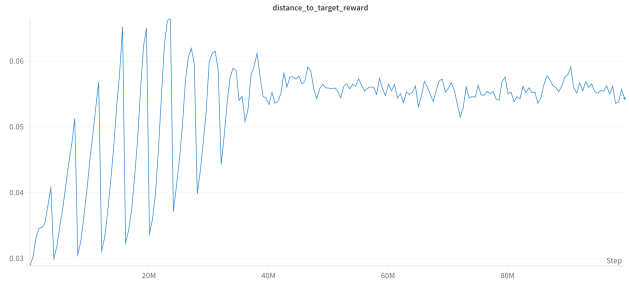

Fig. 15: Dist to target reward per step