Leveraging LLMs for Mission Planning in Precision Agriculture

Marcos Abel Zuzuárregui Stefano Carpin

***Abstract*— Robotics and artificial intelligence hold significant potential for advancing precision agriculture. While robotic systems have been successfully deployed for various tasks, adapting them to perform diverse missions remains challenging, particularly because end users often lack technical expertise. In this paper, we present an end-to-end system that leverages large language models (LLMs), specifically ChatGPT, to enable users to assign complex data collection tasks to autonomous robots using natural language instructions. To enhance reusability, mission plans are encoded using an existing IEEE task specification standard, and are executed on robots via ROS2 nodes that bridge high-level mission descriptions with existing ROS libraries. Through extensive experiments, we highlight the strengths and limitations of LLMs in this context, particularly regarding spatial reasoning and solving complex routing challenges, and show how our proposed implementation overcomes them.**

I. Introduction

Mission planning (MP) can be defined as "any system that plans the operations of another system or any of its components" [15] and is central to the deployment and adoption of autonomous robotic systems operating in semi-structured or unstructured environments [4], [5], [17]. MP is related to other classic areas in robotics and planning, such as task and motion planning [11] and mission specification through linear temporal logic [24]. Solving MP problems involves multiple challenges. One deals with synthesizing a plan leading to the desired outcome. In classic AI research dealing with assembly tasks, it is often assumed that the outcome of actions is predictable, and notwithstanding the problem remains often very difficult due to the large branching factor in the search space. Another aspect relates to *how* one informs the MP system about the desired outcome, especially when the desired behavior involves complex sequences of actions with uncertain outcomes. Our ongoing research in robotics for precision agriculture [9] motivates our interest for MP in this application area, and there are various interrelated challenges we tackle in this paper. As agricultural robots are expected to be operated by non specialists, a natural requirement is to empower users with the ability of synthesizing complex mission plans without having to deal with low level details or understanding system level complexities. Moreover, in precision agriculture missions, robots operate in semi-structured environments where the outcome of actions is unpredictable. Importantly, the outcome of the mission may depend on data collected at run time. For example, a robot may be tasked with collecting pictures of some trees (see e.g. Figure 1), and the actions may be adjusted at run time based on partial results (e.g., if trees appear to be stressed, the robot may be required to collect soil moisture samples). Another layer of complexity arises from the fact that robots operating in rural areas often lack network connectivity, making it impossible to access remote services in real time. Additionally, a critical component of these applications is the need for spatial planning and reasoning, both in quantitative terms (e.g., taking three moisture measurements 15 meters apart) and qualitative terms (e.g., measuring soil nitrate levels on the east side of the orchard).

Fig. 1: A task in precision agriculture: a robot acquiring thermal imaging of a pistachio tree for water stress detection.

Starting from these premises, we present an end-to-end system that tackles these challenges exploiting the novel abilities offered by large language models (LLMs). With the advent LLMs, the intersection between natural language processing, planning and robotics is seeing tremendous growth [18], [23], in an effort to make human-robot interactions seamless. Considering our focus on precision agriculture, we question whether the same principles applied in large language models (LLMs) could be relevant in this domain. For instance, when planning a robot's path through an orchard, numerous contingencies must be taken into account, which can influence the trajectory. While the primary goals are being met, it is also essential to optimize resources and adhere to various constraints. *Can current LLM models solve mission planning problems in precision agriculture requiring to handle uncertain outcomes, spatial awareness and resource optimization?*

In this paper, we examine the use of LLMs to generate

The authors are with the Department of Computer Science and Engineering, University of California, Merced, CA, USA. This work is partially supported by the IoT4Ag Engineering Research Center funded by the National Science Foundation (NSF) under NSF Cooperative Agreement Number EEC-1941529. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the view of the National Science Foundation.

mission plans and explore the standardization of LLM-generated mission plan outputs based on the recently published IEEE Standard for task specification. [2]. We will also show the limitations of ChatGPT and the necessity to augment a mission plan with human designed components to solve mission relevant resource optimization problems. The contribution of this paper are the following:

- we present an LLM to robotic task execution pipeline for autonomous navigation and data collection;
- we show how our use of a LLM mission planner can be integrated with the IEEE standard 1872.1-2024 [2];
- we investigate whether mission plans generated leveraging LLMs can effectively reason about space and consider stochastic optimization constraints typical of the precision agriculture domain;
- we validate our proposed system in the field and show limits and strengths.

The rest of the paper is as follows. Selected related work is presented in Section II. In Section III where we describe the system we developed, experiments detailing our findings are given in Section IV, and conclusions are given in Section V.

II. Related Literature

1) Mission Planning: Literature in MP for robotics is vast, and over the years we have seen an increase in mission planning literature ranging from distributed and dynamic mission planning [5], [17] to applied mission planning [4] to the ever popular manipulator planning [3], [12], [13]. While often mission plans are specified using linear temporal logic (LTL) [16], [17], [19], this creates a difficult interface for end users. Notably, many of these papers focus on some form of runnable code generation. We differ by implementing a well-formed, decomposed task representation by way of [2] to standardize our mission plan output. For example, [18], [23], [25] used logical, non-standardized task decomposition definitions generated in Python. To our knowledge, this paper is the first to use this type of generic non-executable framework in mission planning.

2) LLMs: While solving MP problems, many authors also integrate modern LLMs into their architectures, such as [12], [18], [21], [23], among others. However, in all of these papers and many others, the task is typically some form of robotic manipulation. While there exists literature in autonomous navigation [8], [10], [30] using LLMs, to the best of our knowledge, we have yet to read anything about precision agriculture mission planning using LLMs. The key difference in these autonomous navigational papers is that they mostly focus on semantic reasoning, while we address task representation.

Where previously LTL [14] and Planning Domain Definition Language (PDDL) [2] were used extensively in the past to generate mission plans, the difference in our experiments is that we use ChatGPT to generate Extensible Markup Language (XML) against an XML Schema Definition (XSD). While seen before in [27] where LLM output is well-formed by format-restriction prompting, our contribution also incorporates a rationale behind the format through [2].

III. System Architecture and Design

As mentioned in the introduction, one of our design objectives is to align the MP output with the IEEE standard 1872.1-2024 for representing robotic tasks [2]. This choice is motivated by the goal of developing tools that can be easily interfaced with other systems following the same standard. While we considered alternatives such as IEEE 1872.2-2021 [1], we decided on 1872.1-2024 due to its focus on task specific representation. Accordingly, the framework behind our architecture is broken up into the functional software blocks defined in [2]: specification, user, approval, execution, and evaluation (see Figure 2). These five modules are further organized into two levels. Defined in [2], a Level 1 (L1) plan is an abstract mission format representation with a modular decomposition of tasks. An L1 plan specifies *what* should be done, but not *how* it should be done. In our implementation, L1 plans are represented by XML files following an XSD schema encoding the task representation defined in the standard. A Level 2 (L2) plan defines how each of the tasks in an L1 plan can be executed on a specific robot platform.

Core to the proposed approach is our cloud-based user interface powered by OpenAI's GPT-4o which generates the L1 mission plan. In the current implementation, a simple text based interface is exposed for the user to input a mission using natural language. There are no special computational requirements for this module other than internet connectivity and an OpenAI API token. The second part of the pipeline consists of a set of ROS2 nodes processing missions represented as L1 plans, and decoding them into L2 plans and then executing them. Depending on the robot platform used to execute the mission, these will target specific hardware modules. As explained in Section IV, in our case the L2 module targets a ClearPath Husky with a self-developed sensor suite appropriate for the agricultural monitoring tasks we are interested in. We next describe the role of each of the five functional modules.

1) Specification: The *specification* step provides the context necessary for the LLM to solve the MP problem. It is within this context that a mission described in natural language is later converted into an L1 plan encoded in XML. Accordingly, the specification consists of two files. The first is an XSD L1 schema representing both the mission plan format as well as an available atomic action pool for the robot(s) being commanded. While in some literature PDDL is used to to represent this knowledge, we chose XML for its established presence in the software engineering industry. Notably, the first instance of the XSD framework was also created by ChatGPT after being provided with the standard in [2]. Starting from the initial XSD file produced by ChatGPT, we then added the robot task pool and schema level detail to ensure task sequences are encoded as behavior trees. These manual additions were necessary because they were not part of the standard. Moreover, the addition of the robot task pool does require domain knowledge of which capabilities a

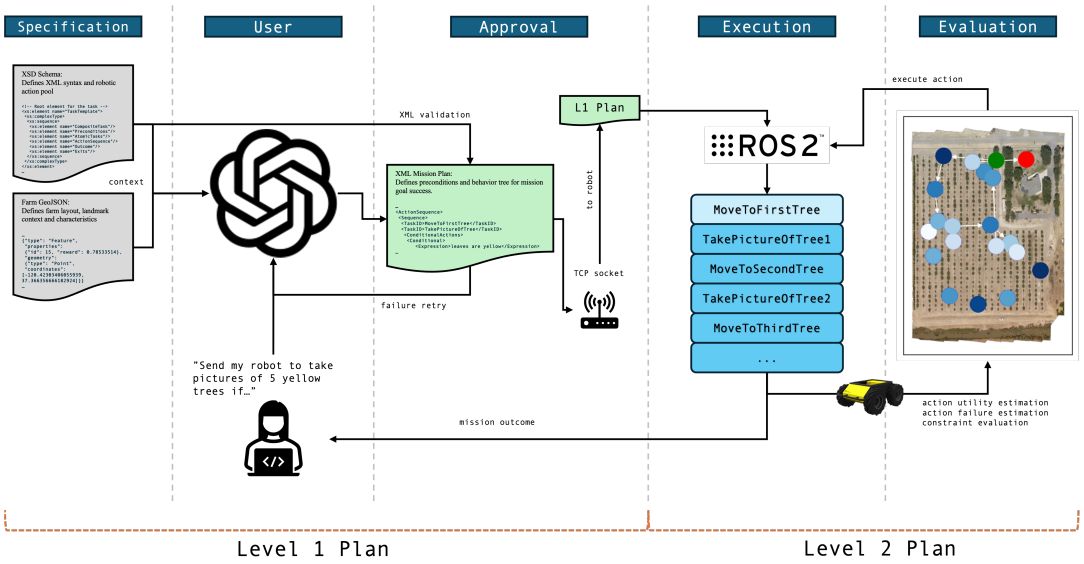

Fig. 2: Our proposed architecture follows the guidance of [2] and breaks up MP into five functional roles. We place each of our software modules into at least one of the five blocks to get a data pipeline that ingests context and natural language, creates an offline L1 plan, converts – at run time – said plan into a L2 plan, and then executes it.

specific robot has.[1] Accordingly, we shaped the relationship between XML tags such that when an L1 plan is generated its resulting task sequence takes the shape of a behavior tree. This creates a conformance to both the IEEE standard and behavior tree modeling, something that can be used generally by other applications seeking to integrate a mission planner. The second part of the specification provides instead the spatial context describing the environment where the mission will be executed. In our domain, this represents the spatial layout of the specific farm where the mission will be executed. This is presented in Geographical JavaScript Object Notation (GeoJSON). This file contains locations and characteristics about the farm, e.g., the GPS location of trees, as well as other characteristics, such as tree attributes (age, type, recorded yield, etc.) and more. In a typical use of our system, a user (described in the next subsection) would start by typing a desired mission query, e.g., "*Send the robot to take pictures of yellow trees in the northern half of the farm.*" GPT-4o translates this request into a mission plan being aware of both the high level capabilities of the robot (as per the XSD file) and the geographic area where the robot should operate (as per the GeoJSON file).

2) User: The *user* role encompasses any entity that utilizes the specification to generate an output. In our implementation, there are two user entities, i.e., the human end user who types the mission request, and the ChatGPT front end that generates an L1 plan. In this context, ChatGPT serves as an extension of the end user, bridging the gap between the user's textual query and the context provided by the specification.

Both user entities bring complementary expertise relevant to the precision agriculture domain. The human, with farming experience, can formulate natural language queries aimed at gathering domain-specific data. ChatGPT, empowered by the background knowledge of the large language model and the provided context, synthesizes mission plans based on underlying robotic capabilities, ultimately achieving the desired outcome – even if the human has limited knowledge of the robotic platform and its capabilities. This approach abstracts from the complexities of MP by allowing ChatGPT to take on the role of curator to eliminate the need for the human to require much context at all, farm or robot related.

3) Approval: After an L1 plan is generated, the *approval* step validates the L1 output against the XSD schema provided in the specification. This ensures that the mission plans can be sent offline to the robot for execution. During this phase, there is a closed loop with the ChatGPT agent ensuring that whatever mission is generated is syntactically valid. This step is driven by the observation that ChatGPT sometimes generates XML files that do not fully comply with the XSD provided in the specification. When this happens, the L1 plan is sent back to ChatGPT together with the error identified by the XSD validator and ChatGPT is asked to fix the error. In all instances we observed that a single iteration through this feedback loop is sufficient to fix any detected error. It shall be noted that this iteration, when executed, is fully automated and does note necessitate of any human intervention. After this first approval step, the L1 plan is sent to the robot, where another validation step occurs on target, to verify successful transmission. This step is critical in that after this stage there is no further syntactic validation and the robot will be subject to whatever mission plan has been generated. After these two validation steps, starting from a simple mission prompt provided by the human end user, a valid L1 plan encoded in XML and compliant with the 1872.1-2024 standard and the XSD schema has reached

[1]The complete XSD schema, supplementary material, and the code are available on `ucmercedrobotics.github.io/gpt-mp-ros2`.

the robot. On the target robot, a set of ROS2 nodes then converts an approved L1 plan into an L2 plan by parsing the XML file and decoding it to generate a task sequence. More specifically, the ROS2 nodes ensure that the L1 input follows the structure of a behavior tree, with attributes that define both each individual task and the overall system state. While this step could in principle also be performed off the robot, in our implementation it is executed on the robot, because verifying robot configuration and preconditions can be done directly while decomposing tasks. After decoding is successful, a list of task objects is generated in ROS2's messaging framework via a service. The robot will first check all preconditions to ensure it is ready to accept this plan. Then, a list of tasks in the form of a behavior tree are parsed and sequenced for the next block.

4) Execution: During *execution*, the robot performs the sequence of tasks derived from the L2 plan with the objective of achieving the goal specified by the human. At this stage, the execution becomes a classic closed loop system until successful completion of the mission. Using the same example query from Section III-.1, a decomposed atomic L2 task would be presented to the navigational ROS2 software stack, Nav2, and the robot would head north. Note that it is up to ChatGPT to extract from the GeoJSON file a GPS coordinate corresponding to "north" and making it available to Nav2 via the pipeline we discussed so far. The handshake between execution and evaluation (discussed in the next subsection) is shared by Nav2 and our custom ROS2 task orchestration node. Both will manage task execution and evaluation to varying degrees. Given execution and evaluation are tightly coupled, we will decouple the data flow and discuss the former in the remainder of this section with the latter in the next. For navigation specific tasks, integral to applications in robotics for precision agriculture, Nav2 is leveraged to ensure accurate execution via localization relying on the Kalman filter node provided by ROS2 augmented with RTK GPS. In our example case, the request to take pictures in the northern half of the orchard will result in a task moving the robot to a specific GPS point satisfying the geographic request. Nav2 handles the navigation, together with obstacle avoidance also leveraging our formerly developed custom extension for robust navigation in agricultural environments [26]. For data collection and sensor tasks, we leverage peripheral management nodes that orchestrates the onboard sensors set to collect data. Following again the same example query from Section III-.1, this node detects yellow leaves via a camera. For each peripheral onboard the robot, a ROS2 service is executed to handle requests as the tasks come in.

5) Evaluation: As the robot executes L2 tasks, *evaluation* is implemented on two different levels. First, we leverage ROS2's Nav2 stack for handling dynamic evaluation of an initial path between two points. The path navigator packaged with Nav2 will receive said points and perform the evaluation of success via the same localization techniques that helped it traverse the path. In the second level, we incorporate custom ROS2 nodes that query data collection for use in adhering to constraints or abiding by the behavior tree. In our example, the robot would constantly be checking navigational status while executing movement between two points: the start point and a tree in the north of the farm. Upon reaching the tree, the next sequential task, taking a picture, would be sent to the picture action server and the orchestrator then waits for a response on whether the tree is yellow or not. In the experiments presented in the next section, we show how evaluation also tackles some tasks that the LLMs alone cannot properly solve. In our application domain we have to assume an offline scenario where there is no network connectivity, thus preventing ChatGPT from possibly revising a mission plan that is not progressing as expected. For example, if the user specifies a mission with resources constraints, the evaluation module monitors progress and intervenes if run time execution shows that the constraints will likely be. This is particular important because we observed ChatGPT's limitations in solving optimization tasks and its occasional inability to properly reason about space. Additionally, the user may specify a mission without constraints because it ignores the the limitations of the robotic system that will execute the mission. For example, robots operating in orchards have limited energy budget and must return to the charging station or deploy point before their batteries are depleted. To complicate matters, the consumed energy is not fully predictable upfront, but is rather a random variable. This problem is known to be an instance of the the stochastic orienteering problem (SOP), a problem we have formerly studied in the context of precision agriculture [6], [28] With a constraint on total distance traveled and no way of pre-planning, due to the stochastic nature of the travel, the ability to manage these constraints at run time is delegated to the evaluation step. It is our contention that the proposed architecture not only solves for SOPs, but can be generalized to solve for other optimization problems that cannot be fully solved offline by the LLM. We will show the results for solving SOPs in Section IV-C.

IV. Results

In this section we show how the system we described can synthesize and execute complex missions starting from high level descriptions in natural language. Due to space limitations, only a subset of results is presented. The companion video, as well as the website linked in Section III-.1, provide additional details and visuals for the interested reader, as well as the code.

A. Experimental Setup

All missions are run on a ClearPath Husky controlled by an Nvidia Jetson Orin Nano running ROS2 Humble. The robot is equipped with a U-blox ZED-F9P GNSS module and with a Bosch BMI085 IMU for localization (both feeding the extended Kalman Filter node part of ROS2 and used for outdoor localization). We used GPT-4o-2024-05-13 with a temperature of 0.2 and set max response tokens to 4096, i.e., the max supported by GPT-4o. While this could be a limitation in that complex mission plans could be longer than 4096 tokens, in our experiments this never happened.

Mission Queries	Number of Tasks	Success?
non-spatial reasoning		
"3 pictures in row of 5"	8 (0 conditionals)	True
"4 trees, 2 sensors"	8 (0 conditionals)	True
"Reward shaping"	20 (0 conditionals)	True
"5 nested if conditionals"	14 (5 conditionals)	True
"If-else with nesting"	16 (5 conditionals)	True
spatial reasoning		
"4 corners relative"	8 (0 conditionals)	False
"4 corners absolute"	8 (0 conditionals)	False
"Sample 100 meters of trees"	10 (0 conditionals)	True*
"North, center, east samples"	6 (0 conditionals)	False
"Relative conditionals"	8 (2 conditionals)	False
"Relative + absolute conditionals"	17 (2 conditionals)	False

TABLE I: Assessment of various mission prompts presented to our MP system. * technically correct, but extremely suboptimal. See Section IV-C for more.

First we assess the ability of ChatGPT to solve MP problems in precision agriculture. We will show how the LLM can generate complex conditional mission plans through the use of behavior trees and the rate of success. We will also show the difference in solving non-spatial and spatial planning problems of various complexities. Based on those initial findings, we will demonstrate a need for additional supportive planning software to optimize LLM based MP. Carried out by our Husky at our testbed, we will demonstrate solving a real-world precision agriculture problem with and without additional planning software. Last, we will assess how simply we can generalize ChatGPT output to solve more than one type of problem. All of these experiments originate from mission prompts that can be found in full on the website and on the video, while Table I shows shortened versions of the full prompts, which do not fit in the table. With these experiments we aim to answer the following questions: 1. *Can we use ChatGPT to solve offline MP problems without any supporting planning software modules?* 2. *Can ChatGPT understand spatially complex queries?* 3. *Can we leverage the generality of LLMs to include optimization and constraints into mission planning problems?*

B. Solving MP problems and Spatial Awareness

We began by evaluating ChatGPT's capability in solving general MP problems with and without spatial awareness requirements. Similar to previous works [18], [22], [23], our goal is to assess ChatGPT's proficiency in generating behavior trees that support mission specifications involving navigation and data acquisition tasks. To this end, we asked an array of mission queries to understand ChatGPT's limitations with respect to conditional complexity and spatial awareness. The set of queries range from simple, complex, conditional, to spatial. Note that all missions were manually reviewed for semantic success. Their results are found in Table I. Syntactically, all missions generated from ChatGPT were correct and paths generated were feasible. When dealing with all forms of non-spatial MP (i.e., requests not asking to reason about space), our architecture never failed. From simple tasks like taking three pictures in a row of five trees to more complex conditional tasks like taking samples of trees depending on the outcomes of previous samples, ChatGPT proved to generate semantically valid and feasible mission plans. However, while we were able to generate behavior tree logic for complex conditional navigational missions, doing so sometimes required an extremely detailed specification in natural language – something a non-specialist user would not easily produce. More precisely, an example request was, "*From the starting point, find a tree somewhere in the middle of the orchard and measure CO_2. If low reading, go to and take 2 pictures of nearby trees. From there, measure another tree as far away as possible for CO_2 and repeat the same process. Once done, take a temperature reading at any one of the northern most trees. Finally, return to end.*" The problem comes from the second sentence, where we had originally intended that the robot visits two unique trees nearby, with "trees" being plural, and take a single picture at each of them. However, no matter the configuration, the LLM produced a mission plan to visit a single tree and take two pictures of it. After changing the second sentence to, "*If low reading, go to 2 different trees nearby and take a picture of each of them*" we finally saw the intended output. The nature of this problem is visualized in Figure 3. In requiring this explicit distinction, we see the need for effective functional validation practices shown by [14] and the venue for future work.

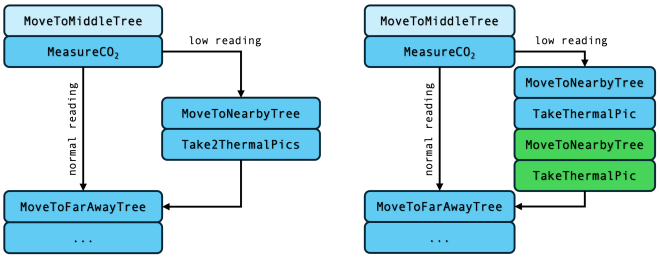

Fig. 3: Demonstration that without explicit intent, sometimes nuanced statements are missed. Here, we see two different behavior trees from two very similar queries. The added tasks are highlighted in green with the starting tasks in light blue.

Next, we tested our system on more complex, realistic missions that would come from a typical data aggregation expedition. This often involves some form of spatial awareness or reasoning, and we correspondingly broke this assessment into two parts: absolute and relative spatial cognition. In these cases we tested ChatGPT's ability to interpret prompt including terms such as "north" or "south" and simple spatial concepts like what is the shape of the orchard in which the robot operates. During this testing, as Table I suggests, we almost immediately ran into problems. While all of the missions were syntactically valid, ChatGPT often times misunderstood the geometry of the farm or its orientation. In two of the queries, "*4 corners absolute/relative*" in Table I, we asked our LLM planner to trace the four corners of our farm plot. As seen in Figure 4, ChatGPT creates a path through the orchard which is close but not actually what we asked for. A similar output in "*Sample 100 meters of trees*", where we ask ChatGPT to sample the most trees possible in 100 meters and it only samples roughly trees spanning 17 meters which is is obviously suboptimal. These nuances

demonstrate the limitations of using only ChatGPT output to guide a navigational stack. In the next subsection, we will expand on this last mission prompt and show how adding a spatially aware software module as part of the evaluation block can immensely improve performance.

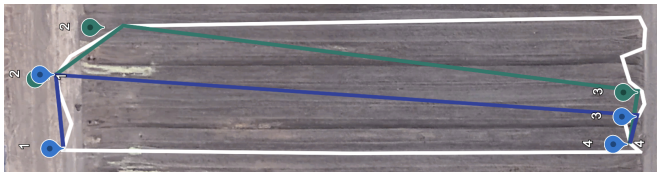

Fig. 4: With left being north, we asked ChatGPT to trace the 4 corners of our "rectangular" farm layout, shown in white. The green is a relative query (corners of rectangle) and the blue is an absolute query (cardinal direction).

C. Optimization and constraints

As we just saw, the current version of ChatGPT does not display the type of spatial awareness needed for MP in precision agriculture [20]. Therefore, given that many of the problems we study have an inherent optimization component related to spatial awareness, we included in our system a spatial planning module to overcome ChatGPT's current limitations. For these experiments, we follow the architecture drawn in Figure 2 and add to the validation block a stochastic orienteering solver [31] to enable constrained route optimization. We focused on stochastic orienteering – a variant of deterministic orienteering particularly relevant in precision agriculture– in which an agent must maximize the utility collected by visiting a set of locations while subject to a hard bound on the traveled distance due to the limited energy provided by the onboard battery. Uniquely, distance traveled is not known until after path execution as the environment is, as the name suggests, stochastic. Performance is measured in two ways: collected utility R, and failure, F, defined as the fraction of missions that violate the constraints. We use these metrics to compare the offline mission provided by ChatGPT used alone or in conjunction with an optimization algorithm (called GPT-SOP in the following) as part of the evaluation block. For completeness

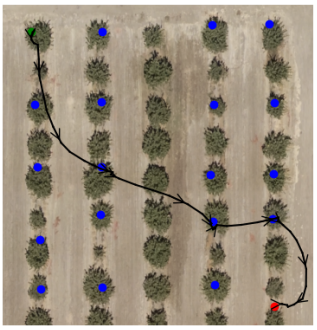

Fig. 5: A plot of our Husky robot's GPS path on our pistachio orchard testbed while solving for an instance of a mission plan where not all trees can be visited with the allocated budget.

	GPT-SOP Solver		ChatGPT		MILP	
	R	F	R	F	R	F
graph20$_{B=2}$	2.092	11%	2.095	4%	3.414	12%
graph30$_{B=2}$	5.592	9%	2.531	10%	6.973	10%
graph40$_{B=2}$	5.785	7%	2.710	13%	8.843	10%

TABLE II: Overall results from benchmarks used in [7] averaged over 100 trials.

we compare also against another offline optimal method based on mixed integer linear programming [29] that is however extremely time consuming (referred to as MILP in the following). This gives us measurable metrics showing how our SOP online solver is essential to the architecture. To grasp the significance of this addition, we started by asking ChatGPT to collect pictures of a large set of trees in an orchard, ignoring the fact that the robot does not have sufficient autonomy to visit them all. Figure 5 shows that even though ChatGPT produces an L1 plan aiming at visiting numerous trees (marked in blue) the GPT-SOP component embedded in the evaluation plan intervenes and limits the visit to a few, ensuring the robot reaches the end point (red dot) before it runs out of energy.

In Table II, instead, we show numerical results where Chat-GPT is asked to produce plans visiting as many trees as possible on orchards with different number of trees (varying from 20 to 40) while sticking to a limit on the maximum traveled distance (parameter B). As we can see, ChatGPT alone (middle), produces plans that are extremely conservative in terms of collected rewards R, while still incurring in relatively high failure rates. Once the system is augmented with our online GPT-SOP solver (left), results dramatically improve. For reference, we also show the results with the optimal offline MILP solver, that however takes extremely long times to compute a solution. When reading the rationale ChatGPT sends with the mission plan, it expresses that it is using some form of a greedy strategy when selecting nodes. Due to the complexity presented in SOPs, such a simple heuristic will only perform so well.

V. Conclusions and Future Work

In this paper we presented a full pipeline that leveraging LLMs starts from a high level objective specified in natural language and produces mission plans that can be executed by a robot operating in an orchard with no internet connectivity. To the best of our knowledge, this is the first example leveraging LLMs in this domain. The solution we proposed provides a valuable tool to enable non specialists to utilize robotics and AI in precision agriculture. Our experimentation has shown that LLM can be beneficial, but may fall short when facing tasks requiring spatial awareness or constrained optimization. These limitations are overcome by augmenting the system with optimization algorithms part of the evaluation block in the proposed architecture. Future research will explore how this architecture can be extended to systems featuring multiple robots, as well as robots interacting with fixed deployed infrastructure such as static sensors.

REFERENCES

[1] IEEE Standard for Autonomous Robotics (AuR) Ontology. *IEEE Std 1872.2-2021*, pages 1–49, 2022.

[2] IEEE Standard for Robot Task Representation. *IEEE Std 1872.1-2024*, pages 1–32, 2024.

[3] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, and C. et al. Finn. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances, August 2022. arXiv:2204.01691 [cs].

[4] M. Askarpour, C. Menghi, G. Belli, M.M. Bersani, and P. Pelliccione. Mind the gap: Robotic Mission Planning Meets Software Engineering. In *Proceedings of the 8th International Conference on Formal Methods in Software Engineering*, FormaliSE '20, pages 55–65, New York, NY, USA, October 2020. Association for Computing Machinery.

[5] B.L. Brumitt and A. Stentz. Dynamic mission planning for multiple mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2396–2401 vol.3, 1996.

[6] S. Carpin. Solving stochastic orienteering problems with chance constraints using monte carlo tree search. *IEEE Transactions on Automation Science and Engineering*, (Accepted for publication) Preprint available on ArXiv https://arxiv.org/abs/2409.03170.

[7] S. Carpin and T. C. Thayer. Solving stochastic orienteering problems with chance constraints using monte carlo tree search. In *Proceedings of the IEEE International Conference on Automation Science and Engineering*, pages 1170–1177, 2022.

[8] L. Chen, O. Sinavski, J. Hünermann, A. Karnsund, A.J. Willmott, D. Birch, D. Maund, and J. Shotton. Driving with LLMs: Fusing Object-Level Vector Modality for Explainable Autonomous Driving. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 14093–14100, 2024.

[9] A. Dechemi, D. Chatziparaschis, J. Chen, M. Campbell, A. Shamshirgaran, C. Mucchiani, A. Roy-Chowdhury, S. Carpin, and K. Karydis. Robotic assessment of a crop's need for watering. *IEEE Robotics and Automation Magazine*, 30(4):52 – 67, 2023.

[10] A. Elhafsi, R. Sinha, C. Agia, E. Schmerling, I. Nesnas, and M. Pavone. Semantic Anomaly Detection with Large Language Models, September 2023. arXiv:2305.11307 [cs].

[11] C.R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L.P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:265–293, 2021.

[12] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents, March 2022. arXiv:2201.07207 [cs].

[13] W. Huang, F. Xia, and T. et al. Xiao. Inner Monologue: Embodied Reasoning through Planning with Language Models, July 2022. arXiv:2207.05608 [cs].

[14] C. Janßen, C. Richter, and H. Wehrheim. Can ChatGPT support software verification?, November 2023. arXiv:2311.02433 [cs].

[15] M. Jones, E. M. Sorensen, T. Wolff, and C. R. Haddow. A review of mission planning systems. In *Proceedings of the Second International Symposium on Ground Data Systems for Space Mission Operations*, 1993.

[16] S. Kalluraya, G. J. Pappas, and Y. Kantaros. Resilient Temporal Logic Planning in the Presence of Robot Failures, October 2023. arXiv:2305.05485 [cs].

[17] S. Kalluraya, G.J. Pappas, and Y. Kantaros. Multi-robot Mission Planning in Dynamic Semantic Environments, March 2023. arXiv:2209.06323 [cs, eess].

[18] S.S. Kannan, V.L.N. Venkatesh, and B.C. Min. SMART-LLM: Smart Multi-Agent Robot Task Planning using Large Language Models, March 2024. arXiv:2309.10062 [cs].

[19] A. Kumar and R. Kala. Linear Temporal Logic-based Mission Planning. *International Journal of Interactive Multimedia and Artificial Intelligence*, 3(7):32, 2016.

[20] F. Li, D.C. Hogg, and A.G. Cohn. Advancing Spatial Reasoning in Large Language Models: An In-Depth Evaluation and Enhancement Using the StepGame Benchmark, January 2024.

[21] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as Policies: Language Model Programs for Embodied Control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 9493–9500, 2023.

[22] A. Macaluso, N. Cote, and S. Chitta. Toward Automated Programming for Robotic Assembly Using ChatGPT, May 2024. arXiv:2405.08216 [cs].

[23] C.E. Mower, Y. Wan, H. Yu, A. Grosnit, J. Gonzalez-Billandon, and M. et al. Zimmer. ROS-LLM: A ROS framework for embodied AI with task feedback and structured reasoning, July 2024. arXiv:2406.19741 [cs].

[24] E. Plaku and S. Karaman. Motion planning with temporal-logic specifications: Progress and challenges. *AI communications*, 29(1):151–162, 2016.

[25] Y. Rizk, M. Awad, and E.W. Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Comput. Surv.*, 52(2), apr 2019.

[26] E. Sani, A. Sgorbissa, and S. Carpin. Improving the ros 2 navigation stack with real-time local costmap updates for agricultural applications. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 17701–17707, 2024.

[27] Z.R. Tam, C.K. Wu, Y.L. Tsai, C.Y. Lin, H.Y. Lee, and Y.N. Chen. Let Me Speak Freely? A Study on the Impact of Format Restrictions on Performance of Large Language Models, August 2024. arXiv:2408.02442 [cs] version: 1.

[28] T.C. Thayer and S. Carpin. A fast algorithm for stochastic orienteering with chance constraints. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7961–7968, 2021.

[29] P. Varakantham and A. Kumar. Optimization approaches for solving chance constrained stochastic orienteering problems. In Patrice Perny, Marc Pirlot, and Alexis Tsoukiàs, editors, *Algorithmic Decision Theory*, pages 387–398, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[30] Z. Xu, Y. Zhang, E. Xie, Z. Zhao, Y. Guo, K.K. Wong, Z. Li, and H. Zhao. DriveGPT4: Interpretable End-to-end Autonomous Driving via Large Language Model, March 2024. arXiv:2310.01412 [cs] version: 4.

[31] M.A. Zuzuárregui and S. Carpin. Solving Stochastic Orienteering Problems with Chance Constraints Using a GNN Powered Monte Carlo Tree Search, September 2024. arXiv:2409.04653 [cs].