An End-to-End Software Architecture Integrating Natural Language Mission Planning and Behavior Trees for Precision Agriculture

Marcos A. Zuzuarregui* **Ali Abedi†** **Reza Ehsani†** **Stefano Carpin***

[1]Department of Computer Science and Engineering, University of California, Merced, CA, 95343, USA
[2]Department of Mechanical and Aerospace Engineering, University of California, Merced, CA, 95343, USA

Corresponding author: Stefano Carpin (email: scarpin@ucmerced.edu).

This work is partially supported by the National Science Foundation (NSF) under Cooperative Agreement EEC-1941529. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the NSF.

ABSTRACT

Deploying autonomous robots in precision agriculture requires systems capable of navigating and interact with partially structured, dynamic environments. Large Language Models (LLMs) are emerging as a powerful interface enabling field operators to repurpose mobile robots through natural language (NL). However, they lack the ability to robustly operate in environments such as commercial orchards. This requires onboard systems to manage perceptive feedback as missions unfold. Furthermore, relying on predetermined large static waypoint databases limits scalability, can exceed effective LLM context windows, and decreases LLM performance. This paper presents an end-to-end distributed software architecture that bridges high-level LLM mission planning with feedback driven robot mission control. The system operates by translating NL requests into structured XML mission specifications, which are transferred to the robot and parsed into fault-tolerant Behavior Trees (BTs). To optimize and scale mission generation, we introduce an orchard management (OM) abstraction layer that interpolates navigation targets from user-defined GPS polygons, thus improving token efficiency. Built on the Robot Operation System (ROS2), the architecture decouples mission orchestration from hardware execution, leveraging Nav2 for mobility, MoveIt2 for manipulation, and dedicated nodes for 3D LiDAR and RGB-D perception. The BT execution engine dynamically manages runtime failures through integrated retry logic, ensuring a robust transition between waypoint navigation and perception-driven canopy approaches. Field experiments conducted in commercial pistachio and citrus orchards validate the architecture's effectiveness, demonstrating that integrating deliberative LLM planning with reactive BT execution significantly improves the reliability, fault tolerance, and scalability of autonomous data collection systems operating in the wild.

INDEX TERMS mission planning, large language models, navigation, precision agriculture

I. Introduction

PRECISION agriculture is one of the fastest-growing application areas for autonomous robotics, including both ground vehicles and aerial drones. It has attracted increasing attention in recent years, driven by rising economic pressure to ensure sustainable and efficient food production across many countries. Quoting [1], *"precision agriculture is the collection of hardware and software technologies that allow a farmer to make informed, differentiated decisions regarding agricultural operations such as planting, fertilizing, pest control, and harvesting."* Precision agriculture is an inherently data-driven area, without which farmers cannot proactively respond to challenges such as managing crop variability and optimizing the use of resources [2]. As such, robotic systems play a key role in promoting sustainability and profitability through the automation of repetitive tasks

FIGURE 1. A Bonsai's Amiga mobile robot operating in a pistachio orchard showing row-based waypoint movement. While navigating through the waypoints, the robot must remain aware of the spatial variability associated with the canopies of the trees.

involving sensing, actuation, or a combination of the two. However, unlike many indoor robotic operational environments, precision agriculture often requires robots to operate for extended periods of time in environments that are only *partially structured*. Orchards are partially structured in the sense that while many farms today are almost algorithmically laid out, with trees often arranged on regular grids with centimeter-level accuracy, plant growth (e.g., canopy structure) remains dynamic and stochastic. For example, a farmer might plant a row of trees facing north on exactly one acre with fixed spacing, but to autonomously collect data, a robot must consider not only the known location of each tree, but also how branches have grown, where leaves are clustered, and where fruit is hidden, among other factors.

Our research in this domain is motivated by the need to deploy ground robots, such as the one shown in Figure 1, to autonomously map orchards and perform proximal sensing to generate high-resolution snapshots of orchard health. This capability is critical for enabling precision management strategies informed by plant-level measurements, supporting early detection of stressors and more targeted interventions at the tree scale. Our system's successful deployment enables large scale, autonomous data collection for all types of proximal sensing where alternative methods may be costly or inefficient.

Robotic systems deployed to support this type of precision agriculture task must feature a perception pipeline that enables them to operate successfully in these challenging environments, ideally without human supervision. An additional challenge to the large-scale adoption of autonomous robots in this domain lies in their user interfaces and the need for them to be easily repurposed, given the diversity of tasks associated with farm operations. Since end users are most often agricultural experts without extensive robotics expertise, it is imperative that these systems be accessible and reprogrammable through intuitive interfaces, with reprogramming often taking place in the field itself. Recently, the use of large language models (LLMs) and vision language models (VLMs) has emerged as a potential approach to address this usability challenge [3], [4] by allowing end users to interact with complex robotic systems using natural language (NL).

Today, there are a growing number of solutions leveraging computer vision (CV) that address the types of perception-based problems that typically arise in precision agriculture, such as segmentation [5], fruit detection and grasping [6]–[9], pruning [10], [11], and proximal sensing [12]. However, in such environments, deploying an autonomous CV-based data collection pipeline driven by NL robotic mission planning remains a significantly more challenging task. Mission planning (MP) in this context is defined as *"any system that plans the operations of another system or any of its components"* [13] and is critical to the deployment and adoption of autonomous systems operating in the same partially structured environments described above [14]–[16]. LLMs are becoming a staple for NL processing of user inputs to generate robot mission plans [17]–[19]. These language models act as a generic interface for users to express their intent and decompose the mission input into a formalized specification. In works such as [17], mission planning powered by LLMs has been shown to be generic enough to be used in any type of environment, but the plan itself remains high-level and offline, as certain information about the mission will not be available until runtime.

In our target application, if a robot is tasked with collecting orchard data and must interact with the leaves to do so, the LLM has only limited context for decomposing the mission with respect to the specifics of each subtask. This means that while a planner may provide a GPS location for a tree, it remains unclear how to abstractly express the interaction strategy such that the robot can robustly approach and manipulate foliage in a partially structured and possibly dynamic tree canopy. If we use a static approach-distance policy from the tree trunk, the robot might reach a vacant spot adjacent to the tree, but the manipulation action may be unreliable due to variations in tree canopy volume. This type of interaction requires to augment the offline, high-level LLM-based MP with continuous task-level feedback-loop integration of perception, planning, and control to ultimately achieve a single goal.

Accordingly, we study the problem of autonomous proximal leaf interaction as the integration of high-level LLM-based MP. Specifically, it concerns the challenge of identifying, selecting, and physically interacting with suitable leaf targets under partial observability, occlusion, and dynamic environmental variability. Unlike traditional manipulation tasks that assume structured geometries or predefined policy-

based approaches, this problem requires continuous fusion of 3D perception, semantic segmentation, and reactive motion planning to ensure both reachability and task reliability. Despite recent advances in NL-based robotic mission planning and perception-driven agricultural robotics, there remains a critical gap between high-level task specification and low-level physical interaction in partially structured outdoor environments. Our main contribution is the implementation of an end-to-end system that seamlessly integrates high-level planning with low-level control through a generic architecture that can be adapted to a variety of tasks. Most importantly, this architecture is demonstrated to be field ready with considerations for fault tolerance and the partially structured nature of proximal sensing.

In this scenario, a robust MP pipeline is required to handle an NL mission provided by the operator at initialization, as well as runtime perceptions during task execution. We therefore propose an end-to-end architecture for an NL-based mission planner capable of partially structured orchard navigation via 3D LiDAR and proximal sensing using a robotic arm. In this paper, we focus on two distinct modules within our proposed architecture: navigation in partially structured environments via NL MP and CV-based manipulation. The contributions of this paper are the following:

- we propose a software architecture[1] that integrates high-level LLM based mission planning with closed-loop task execution.
- we demonstrate how this architecture can be instantiated to solve any large scale proximal sensing problem for high-value perennial trees;
- we demonstrate the effectiveness of the proposed system through proximal leaf sensing performed in commercial orchards growing different trees.

The rest of the paper is as follows. Selected related work is presented in Section II. Section III provides a brief overview of the robotic system we use to collect samples in the field. In Section IV we describe the system, with experiments detailing our findings are given in Section V and we discuss them in Section VI. Finally, conclusions are presented in Section VII.

II. Related Work

A. Mission Planning with LLMs

The integration of LLMs into robotic MP has evolved from basic NL interfaces to complex hierarchical systems. While early implementations demonstrated success in translating user intent into executable scripts [20], [21], many of these systems suffered from lack of verifiable outputs or brittle system execution [22]. To mitigate these risks, recent literature has shifted toward grounded planning [23], where LLMs interface with external validators to ensure reliability in the form of syntax feedback loops [17] and/or formal verification [24]. There has also been exploration into vision language models (VLMs) to overcome deficiencies in real-time reasoning during precision agriculture mission execution [25]. Despite these gains, the "human-in-the-loop" remains a bottleneck for true field autonomy.

Recent advancements have also addressed the challenges of coordinating multiple robots with diverse capabilities. [26] introduced a hierarchical mission-planning strategy that utilizes LLMs to construct task trees, enabling heterogeneous teams to decompose complex objectives. In the specific context of precision agriculture, [27] demonstrates an architecture that translates high-level farm orchard missions into primitives executable by both wheeled mobile platforms and computer-vision-enabled manipulators. Their approach highlights a critical requirement for field robotics, i.e., the ability to operate in one-shot planning modes where persistent network connectivity for human-in-the-loop feedback cannot be guaranteed.

Our work extends these MP paradigms by focusing on the autonomous generation and robust, self-recovering design of mission specifications specifically for a mobile manipulator in agricultural settings. By removing the need for end-users to possess expertise in formal logic while maintaining a self-sustaining feedback loop, we address the unique reliability demands of autonomous field robotics.

B. LLM to Behavior Trees

The application of LLMs to behavior tree (BT) generation has emerged as a promising approach for bridging high-level NL intent with verifiable robot execution. Previous grounded planning work coupled LLMs with external validators [17] or formal verification [28] to provide syntax feedback and safety guarantees. Similarly, [29] grounded LLM reasoning directly into BTs, creating a recoverable link between abstract objectives and low-level execution primitives. BETR-XP-LLM [30] further demonstrated dynamic BT expansion at runtime to resolve execution failures, though evaluations remained confined to controlled manipulation settings.

Fault tolerance and robustness in LLM-driven execution have become key concerns as these non-deterministic systems move toward deployment. BT representations are well-suited to meet this requirement, as their hierarchical and modular structure naturally exposes execution state and supports recovery behaviors. The work presented in this paper advances LLM-to-BT planning by targeting deployment in partially structured agricultural environments where prior systems fall short. Rather than relying on static waypoint databases that strain LLM context windows [31] or simply be cost ineffective, we introduce an orchard abstraction layer (orchard management, OM) that interpolates navigation targets from GPS polygons, improving scalability and mission generation efficiency. By translating natural language requests into structured XML mission specifications parsed into fault-tolerant BTs with integrated retry logic, the architecture bridges deliberative LLM planning with the closed-

[1]The software architecture is available as open source to researchers and practiotioners (URL currently removed to ensure double blind review.)

loop reactive execution required for tasks that may require supervision.

C. Orchard Navigation

Row navigation is a foundational capability for agricultural robots because many field operations require long-distance traversal along plant rows, even when the visual structure is discontinuous or the crop geometry changes. Prior work typically addresses this by combining: (i) row perception from vision, ranging from fast classical line extraction [32] to deep learning segmentation pipelines deployed on embedded platforms [33] and lightweight pruned networks that output a navigation line and yaw via curve fitting [34]; (ii) localization and heading estimation using low-cost multi-sensor fusion and GNSS when available [35], [36]; (iii) fallback or GNSS-denied strategies designed for canopy-induced degradation, including LiDAR SLAM mapping for orchard corridors [37], trunk-structure-based localization with filtering [38], and explicit switching between waypoint tracking and exteroceptive row following to extend autonomy under poor GNSS [39]. More recent systems also integrate multi-camera navigation and mapping without GNSS [40], stereo-vision plus hybrid control for fruit-tree rows [41], LiDAR-first in-row autonomy for under-canopy crops [42]–[44], and planning/exploration layers that improve full-farm coverage and reduce local-optima failures [45], [46], alongside practical field platforms that fuse GNSS, RGB, and multi-line LiDAR with row-based visual correction [47].

A large body of work focuses on vision-based row perception to directly infer a drivable corridor and extract a row centerline. Classical approaches emphasize fast geometric fitting. For example, for greenhouse spraying, a median-point Hough transformation pipeline segments plants vs. soil and fits a navigation path with low latency [32]. More recent work shifts to semantic segmentation for robustness under lighting changes. A comparative study deploys multiple segmentation networks (e.g., UNet/DeepLabv3+/BiSeNet/ENet) on embedded hardware and then fits the navigation line from the segmentation output [33]. Similarly, a pruned "Faster-U-net" uses transfer learning across crops and then generates the navigation line and yaw using B-spline fitting, targeting row/ridge prediction under weeds and light interference [34]. For orchard contexts, stereo vision has also been used to identify trees and maintain row following. One framework combines stereo-camera-based crop perception with a Dynamic Window Approach (DWA)-based controller to follow fruit-tree rows without explicitly switching between row-following and trajectory-tracking modes [41]. Beyond single-sensor perception, camera fusion has been explored for full-field visual navigation. For instance, a ROS-based phenotyping robot fuses multiple cameras to follow rows and transition between them without GNSS is presented in [40].

A complementary direction improves localization and heading estimation to support row-centered tracking over long distances. Modular platforms integrate ROS navigation with GNSS-based localization (including dual-GNSS antennas for heading) and implement local line-following planners in the navigation stack to maintain row traversal accuracy [36]. Cost-effective sensor-fusion is another theme. RoboNav combines dual low-cost GPS receivers with multiple low-cost IMUs using a Gaussian-sum filtering strategy to preserve position and heading estimates under bias and intermittent GPS loss, and validates waypoint tracking in a commercial vineyard [35]. When GNSS quality degrades specifically in orchards, several works fuse vision/odometry with inertial and wheel-related signals. One approach estimates row-relative pose from a monocular camera and then uses a Kalman-filter-based global fusion of vision, RTK-GPS, IMU, and motor Hall sensors, together with preview-based tracking control [48]. A hybrid "switching" strategy for under-canopy robots explicitly alternates between GNSS waypoint tracking and exteroceptive row-following (LiDAR), adding failure detection and recovery to reduce interventions during long field runs [39].

D. Outdoor Navigation

To reduce reliance on GNSS altogether, many systems leverage LiDAR-based structure and SLAM in crops and orchards. For orchard spraying, a 3D LiDAR SLAM pipeline builds a 3D point-cloud map and converts it into a 2D grid representation for navigation and obstacle avoidance is handled within ROS [37]. In orchard rows specifically, 2D LiDAR trunk detection has been used as a stable geometric cue: one method clusters trunk returns using a DBSCAN–ratio–threshold procedure, matches trunk centers to a known tree distribution map, and fuses estimates via an Extended Kalman Filter (EKF) to output robust pose for navigation [38]. In dense row crops, a recent of work develops LiDAR-centric in-row autonomy. The P-AgSLAM system proposes a 3D LiDAR SLAM framework tailored to extract crop-specific morphological features for state estimation and mapping [43]. Building on similar sensing, P-AgNav uses 3D LiDAR range-view images to enable in-row navigation with collision avoidance and row switching without GNSS or pre-defined waypoints [44]. Earlier system work on an under-canopy robot (P-AgBot) demonstrates integrated navigation for monitoring and sampling in corn and sorghum, emphasizing reliable operation inside rows where canopy effects are prominent [42]. Finally, affordability and season-to-season robustness motivate pipelines that combine coarse global representations (e.g., occupancy grids) with learning-based components and synthetic data generation, explicitly targeting GNSS unreliability and vegetation-induced degradation inside rows [49].

Despite substantial progress in agricultural robot navigation, existing systems primarily emphasize row-following behavior, where the robot traverses along the corridor between crop rows while maintaining a stable trajectory relative to the row structure. In this setting, perception, localization, and control are optimized for continuous motion through the

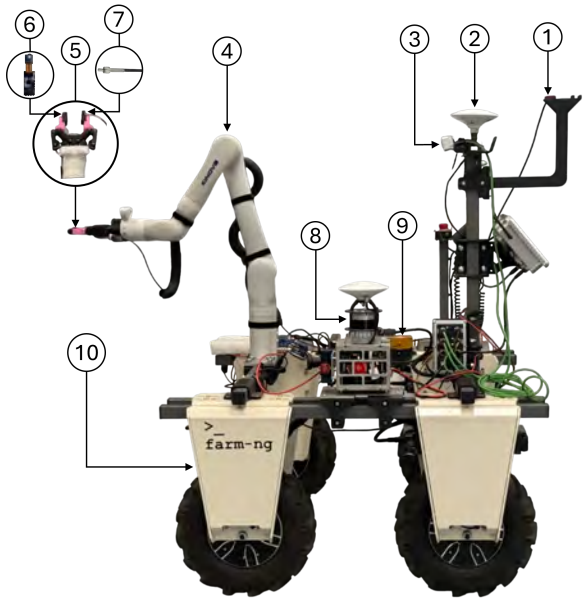


FIGURE 2. The mobile manipulator platform developed to perform water stress detection. 1. IMU sensor; 2. GPS; 3. RGBD camera; 4. Six degree of freedom robotic arm (Kinova Gen3); 5. Two-finger gripper; 6. Nano spectrometer; 7. Fiber-optic cable; 8. 3D LiDAR; 9. VIS-NIR light source; 10. Mobile platform.

row rather than for selecting and approaching an individual plant instance. To the best of our knowledge, prior work has not addressed *tree-specific navigation*, where the robot must leave the nominal row-center trajectory to autonomously approach a designated tree (e.g., for targeted sampling or inspection) and then continue its mission. Motivated by this gap, we propose a tree-specific navigation framework that enables goal-directed approaches to individual trees within orchard rows, extending standard row navigation into a per-tree capability for precision operations.

III. System Overview

To perform the leaf sampling task described in Section I, we developed a field-deployable robotic leaf-sensing platform composed of a mobile base, a manipulation module, and a multi-modal perception and sensing suite for canopy perception and close-range leaf interaction (see Figure 2.)

A. Hardware Platform

The platform is built on a Bonsai (formerly farm-ng) Amiga agricultural mobile robot configured with four wheels that can be independently driven as a 4×4 skid-steer system. A Kinova Gen3 6-DOF robotic arm is mounted on the right side of the mobile platform and equipped with a Robotiq gripper for physical interaction with leaves. Onboard sensing includes an RTK GNSS receiver (Emlid Reach RS2) with an external GNSS antenna and NTRIP-based correction, providing centimeter-level global positioning for navigation and repeatable sampling. A VectorNav VN-100 IMU is mounted on the mobile platform to provide orientation and motion measurements. Visual perception is provided by an OAK-D W PoE stereo camera, connected through a PoE switch to an NVIDIA Jetson AGX Orin compute unit. All primary sensors are interfaced to the Jetson, and all perception and autonomy processing is executed onboard. The Jetson AGX Orin is configured with 64 GB RAM, JetPack 6, Ubuntu 22.04 and using ROS2 Humble, and the Kinova arm communicates with the Jetson via Ethernet.

For precise control during the final approach to a target tree for sampling, 3D LiDAR sensing is required to estimate distance to the canopy and maintain a prescribed safety margin. In our system, LiDAR sensing is provided by a custom developed SensorTower[2] equipped with an Ouster OS0-128-U LiDAR. The SensorTower runs on a separate Intel NUC computer (Ubuntu 20.04) and is networked to the Jetson and provides LiDAR data to the Jetson.

The optical sensing components for leaf sensing is mounted on the gripper and consists of a VIS-NIR hyperspectral spectrometer (NSP32m W1, NanoLambda) operating over 350–1010 nm, together with an independent light source spanning 350–2500 nm to support self-calibration and consistent acquisition under varying ambient illumination. A VIS-NIR fiber optic cable (F600, StellarNet Inc.) routes illumination to the gripper-mounted end-effector region (see also Figure 8.) The spectroscopy sensor is connected to the Jetson via USB. When the robot reaches the target, the gripper captures the leaf and holds it in a fixed pose so the integrated optical head can acquire a repeatable hyperspectral measurement. During the grasp, the spectrometer records the leaf's VIS–NIR spectral response (reported in the 350–1010 nm range) as the hyperspectral sample that is stored for downstream analysis (e.g., phenotyping and, in our application, stress/nutrient assessment). All major subsystems (mobile platform, Jetson, SensorTower, Kinova arm, and the spectroscopy light source) are powered from a shared 44 V, 15 Ah battery through dedicated DC–DC converters to provide the required voltage rails and electrical isolation.

B. Software Architecture

The software architecture governing the hardware platform is comprised of two primary functional blocks (see Figure 3.) The first is a high-level control layer consisting of a web-based MP system that serves as the interface between the user and the robot. This block leverages OpenAI's gpt-5.4-2026-03-05 API with "medium" reasoning to translate objectives expressed in NL into an Extensible Markup Language (XML) file that encodes the mission specification through a BT structured mapping of elementary capabilities offered by the robot, which are described in Table 1.

This component is our updated version of the open-source implementation of the framework described in [17]. Originally, [17] implemented a bespoke BT implementation along with providing all mission input directly to the LLM.

[2]The original sensor tower design is by one of the universities collaborating with us on this project.

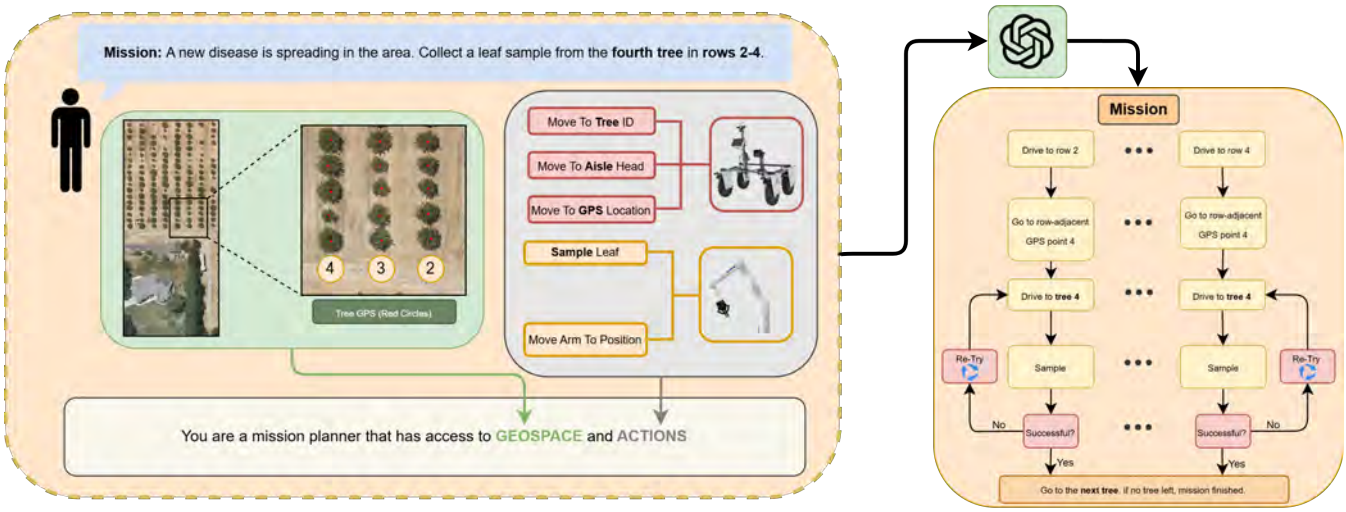


FIGURE 3. **Our mission planning architecture, adapted from [17]. Users provide NL input to request their mission. Ultimately, a BT is executed by a robot in the field.**

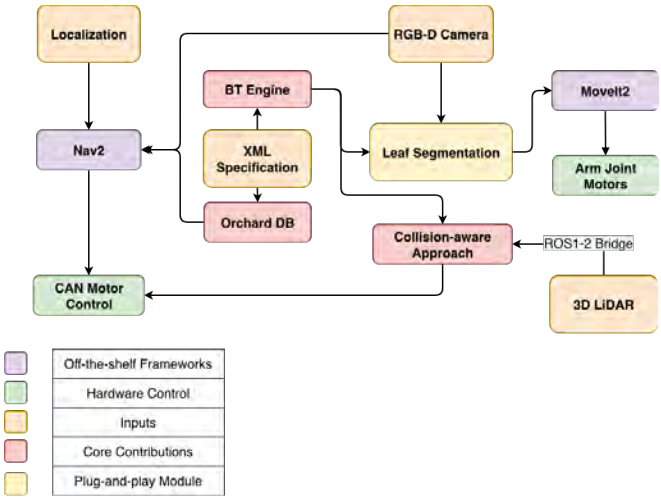


FIGURE 4. **System diagram showcasing the integration between ROS2 provided nodes and the components we developed.**

Further described in Section IV, our updates are to the BT driver, enabling a standardized BT implementation, and to how context is compacted prior to LLM prompting.

In the second functional block, the MP software communicates with the robot platform via TCP to provide a complete mission encoded in XML. Once the mission specification is successfully transferred to the robot, an internet connection is no longer required. This design allows the hardware to operate in network-constrained environments or areas where persistent connectivity is unavailable. The robot's software architecture is implemented using ROS2 Humble, following a distributed design that decouples mission orchestration, navigation, perception, and manipulation. Mission execution is managed by BTs, which serve as the system's high-level task manager. The mission execution engine parses specifications defined in XML to sequence tasks such as autonomous navigation and leaf sampling routines. By utilizing a BT-based approach, the system maintains a clear separation between high-level mission logic and low-level hardware control. The architecture integrates several ROS2 frameworks for autonomous operation. The items below, among this paper's contributions, are captured in the system diagram in Figure 4.

TABLE 1. **Elementary robot capabilities for LLM-generated behavior trees.**

Action	Description	Base	Arm
MoveToTreeID	Navigate to orchard tree.	✓	
MoveToAisleHead	Navigate to aisle entrance.	✓	
MoveToGPSLocation	Navigate to GPS waypoint.	✓	
ApproachGPSWaypoint	GPS navigation with LiDAR approach.	✓	
MoveToRelativeLocation	Base translation in robot frame.	✓	
OrientRobotHeading	Absolute or relative base rotation.	✓	
FollowPerson	Vision-based person following.	✓	
SampleLeaf	Leaf detection, grasp, and sampling.		✓
MoveArmToPosition	Relative end-effector motion.		✓

- Nav2, an autonomous navigation framework, handles global and local path planning, interfacing directly with the Amiga's Remote Procedure Call (gRPC) Controller Area Network (CAN) drivers.
- Motion planning and inverse kinematics for the Kinova Kortex are implemented using MoveIt 2.
- As the 3D LiDAR drivers operate within a legacy ROS1 Noetic environment, a ROS1–ROS2 bridge[3] is utilized. This utility enables the bidirectional exchange of point cloud between legacy hardware and our presented ROS2-driven algorithms.

[3]https://github.com/TommyChangUMD/ros-humble-ros1-bridge-builder.git

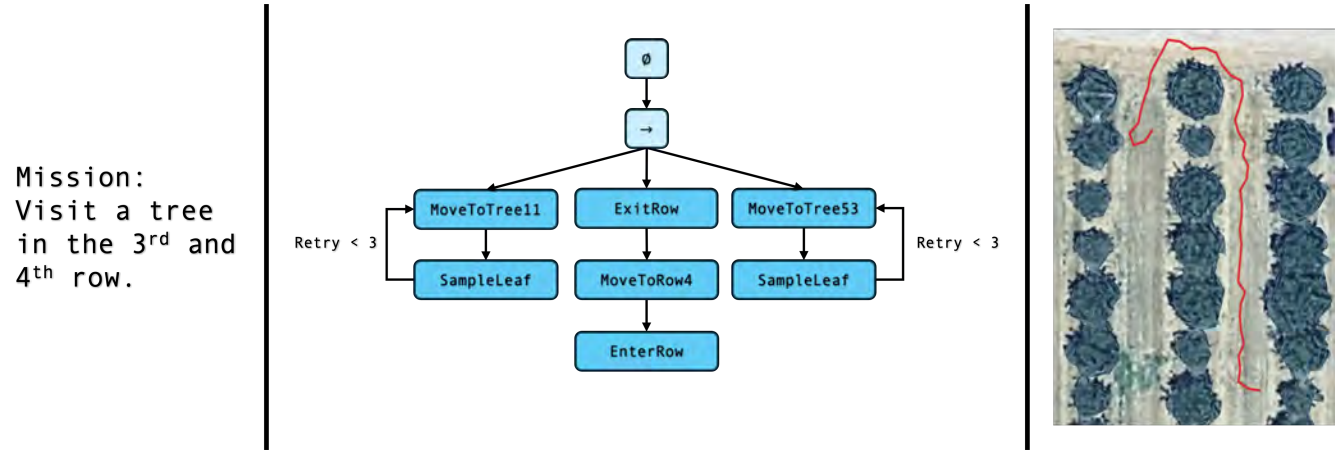


FIGURE 5. **Example multi-tree mission similar to Mission 3 shown in Table 4. On the left, we have a user mission input. The middle is the decomposed BT from the available atomic actions via the LLM planner shown in Table 1. Finally, on the right, the actualized GPS path within the pistachio orchard.**

Perception and orchard management (OM) data are handled as independent nodes within the ROS2 distributed network accessible via topics, actions, and services ensuring that compute-intensive vision tasks do not interfere with the real-time constraints of the navigation and manipulation pipelines.

IV. Methods

A. Mission Planning and Task Orchestration

Our architecture begins with the system proposed in [17] featuring the user interface to creating robot missions. In our case, we use this package to generate four sets of experiments documented in Section V. This LLM-based MP software allows for NL-based missions to be parsed into BTs for the robot to consume. An XML schema definition (XSD) written to match the format of BehaviorTree.CPP (BT.CPP), the default BT of ROS2 [50], is given as context along with the users requested mission. Since the XSD acts as a syntax checker and BT.CPP natively uses XML as input, we can ensure that LLM outputs are always feasible and can be parsed by the BT.CPP engine. Prior implementation considered [51] as a design standard, but did not have an implementation standard. One of our functional updates is in the form of the BT.CPP specific schema that allows LLM output to be readily accepted by the BT.CPP ROS2 engine instead of requiring a custom BT implementation. The reader is also referred to appendix A and the website mentioned there for additional details on the XML schema definition. This XML is then validated on board the robot to be parsed as a BT at runtime. However, [17] is strictly an off-line subsystem that does not account for real-world execution, where missions may encounter failures. Instead, the generated BTs make assumptions about the implied success of tasks within the mission. Completing missions autonomously in a real farm settings requires robust and fault tolerant execution. Therefore, we enhanced the previous mission generation pipeline to support full autonomy in the field by updating the XML schema to support task retries and real-time task evaluations. For more details on the original MP software design, see [17]. Figure 5 shows an example of how this module translates a mission expressed in natural language into a BT whose nodes have a one-to-one correspondence to individual capabilities available on the robot in the form of ROS2 services.

Additionally, the previous system [17] implemented a context file with a full list of all GPS points of interest. This was passed to the LLM planner along with the user's prompt to create a full mission representation. However, in a real deployable setting, having a database of all trees in an orchard or of all points of interest is not always realistic and may moreover create context files with a size that that may exceed the capabilities of the underlying LLM. Instead, we extended the system to include an OM component that estimates tree locations, row entrance and exits, and tree adjacent locations based on a GPS polygon given by the user. This way, instead of requiring tree markings, the user simply gives a polygon reference, enters how many trees are in the orchard (rows by columns), and the OM component interpolates the tree locations based on even spacing as well as each possible row waypoint required to interact with the tree, as visualized in Figure 6. This polygon reference is in the form of GPS coordinates and is represented as the four white dots in Figure 6. In a real scenario a robot would never drive right to the exact GPS location associated with a tree, so this approximate interpolation allows for adjacent waypoints to be computed that enable the robot to get near the tree while using its onboard sensors to keep a safe distance from the canopy. Importantly, while our experimental case is a uniform grid of trees, this logic also works for any polygon with a non-uniform tree distribution per row. By removing the GPS listing from the LLM planner, this frees context space for the LLM to respond more effectively. Starting from the the polygon instead of the tree absolute locations, the planner can then generate missions

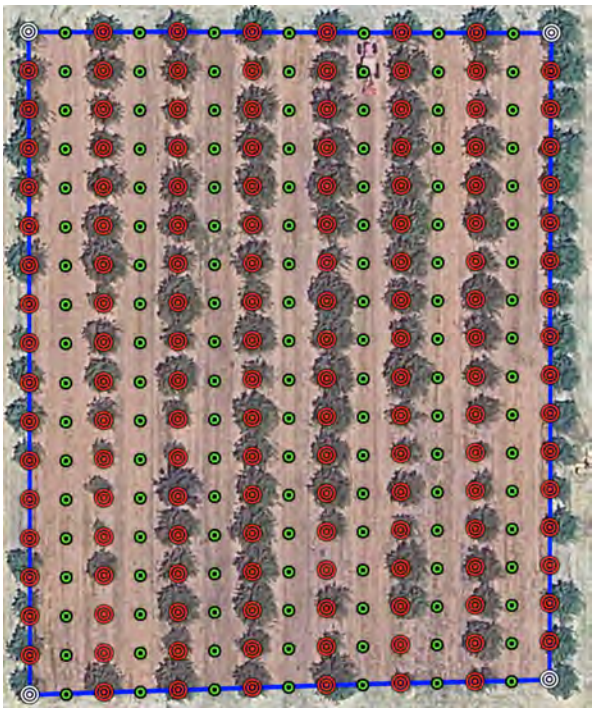


FIGURE 6. **Interpolated GPS points of trees and adjacent row waypoints over a satellite image of our pistachio orchard. Red dots represent tree GPS markings while green are adjacent row-accessible GPS waypoints. The user only provided the white dots as input, along with tree count.**

Algorithm 1: Target-Tree Sampling

Input: User-specified tree estimates $\mathcal{G} = \{\mathbf{p}^1_{\text{tree}}, \ldots, \mathbf{p}^N_{\text{tree}}\}$; attempt budget K; safety distance d; robot pose $\mathbf{p}_{\text{robot}}$, ψ_{robot} from localization

foreach $\mathbf{p}_{tree} \in \mathcal{G}$ **do**
  $\mathbf{g}^{\text{start}} \leftarrow$ AISLESTART($\mathbf{p}_{tree}$)
  GOTOWAYPOINT($\mathbf{g}^{\text{start}}$)
  GOTOWAYPOINT($\mathbf{p}_{tree}$)
  $sampled \leftarrow$ **false**
  $attempts \leftarrow 0$
  while $\neg\, sampled \ \wedge\ attempts < K$ **do**
    $\mathbf{p}^\star \leftarrow$ CLOSESTCANOPY($\mathbf{p}_{tree}$)
    $\phi^\star \leftarrow \operatorname{atan2}\big(p^\star_y,\ p^\star_x\big)$
    $r^\star \leftarrow \sqrt{(p^\star_x)^2 + (p^\star_y)^2}$
    $\ell \leftarrow \max(0,\ r^\star - d)$
    ROTATE($\phi^\star$)
    if $\ell > 0$ **then** DRIVEFORWARD(ℓ)
    $sampled \leftarrow$ SAMPLE()
    $attempts \leftarrow attempts + 1$
  GOTOWAYPOINT($\mathbf{g}^{\text{start}}$)

based on relative locations (e.g., the third tree down the tree row) or estimated GPS positions (right side of the field), but not on exact points of interest. As it will be shown in Section V this significantly increases robustness.

Low-level precision navigation towards specific trees is accomplished by the second layer of the OM software that runs on the robot as a ROS2 node. This node is tasked with parsing through the interpolated tree points and their associated row waypoints to generate a concrete mission based on the abstracted XML version. This BT becomes the robot specification for which it will interact with all ROS2 subsystems, included Nav2. With the XML mission specifying which tree to interact with, the system allows for robot perceptions to drive physical interactions. This saves the user time by reducing system learning curves and initial robot configuration. More importantly, by removing the need for exact precision in orchard layout, the robot can rely on its own sensors to figure out how and where to approach the tree instead of making assumptions in the mission planning stage. This is critical as each tree has its own unique canopy shape and requires a different approach. In the next subsections, we discuss how this abstracted mission plan iterates through a partially structured orchard.

B. Aisle Switching

In the orchards we target, the tree rows form repeated obstacles that constrain where the robot can drive. The local Nav2 planner produces collision-free motion within its costmap, but our forward-facing depth camera only sees a short distance ahead, so the planner never has the orchard's full connectivity available to it. Because of this, the path that is locally shortest toward a target tree frequently points through a tree row instead of down an open aisle, and the robot repeatedly commits to a route it cannot physically follow. The reason becomes clear once each tree is inflated by the robot's radius and a safety margin, $\rho = r_{\text{tree}} + r_{\text{robot}} + r_{\text{safety}}$. When the in-row spacing satisfies $d_{\parallel} < 2\rho$, the inflated trees overlap and the row becomes a solid barrier. The wider between-row spacing $d_{\perp} > 2\rho$ keeps the aisle itself open. For this reason, moving between aisles cannot be handled reliably by minimizing Euclidean distance to the goal on a partial map. In our current implementation we instead decompose the motion into a short sequence of waypoints: the robot exits its current aisle, drives to the entrance of the target aisle, and only then approaches the tree. Providing these waypoints gives the planner the orchard layout it cannot observe on its own, and in practice keeps the robot from wasting time on geometrically short but infeasible paths through the rows. As with the rest of our navigation stack, we kept this deliberately simple, and found it sufficient for the commercial orchards in which we tested. Algorithm 1 summarizes the complete per-tree execution procedure: the outer structure captures the aisle entry and exit waypoints described above, while the inner loop formalizes the canopy approach detailed in the following section.

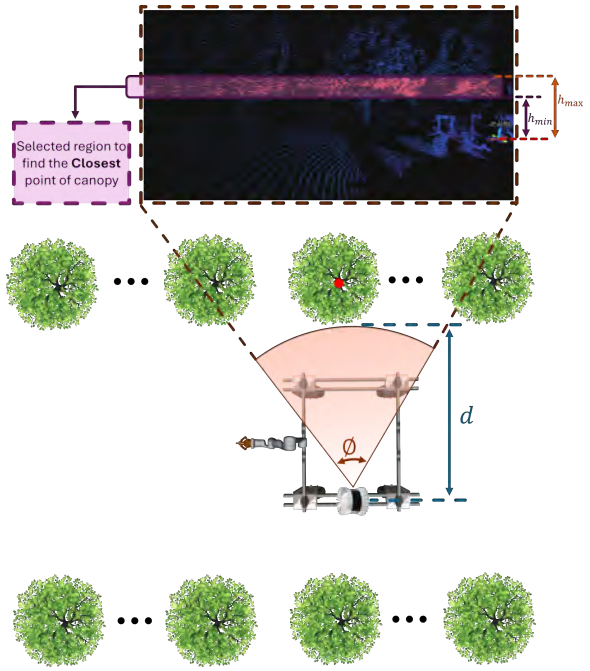


FIGURE 7. High level view of how the robot determines its relative position to the target tree. Highlighted points are closest-point candidates.

C. Inside Canopy Navigation

While the mission plan organizes the offline task structure, task execution requires runtime reactive control. Specifically, when the robot is called to collect samples from a tree, it must be able to do so while accommodating for the unique shape and size of each tree. At this stage the robot knows its current location $\mathbf{p}_{\text{robot}}$ within the orchard (as per the RTK GPS), the estimated tree location $\mathbf{p}_{\text{tree}}$ (as per the interpolation described in the previous subsection), and a 3D LiDAR point cloud of its surroundings. With this information, the robot must identify an accessible subset of the tree canopy and compute a goal position that enables reliable physical interaction while accounting for environmental variability. In an effort to keep the computation simple and robust, in our current implementation the task accomplished by computing the polar coordinates (direction and distance) of the desired point where the robot should position itself to robustly commence the leaf sampling maneuver (see Figure 7 for the remainder of the discussion.)

Let $\mathcal{P}$ be the set of points returned by the 3D LiDAR, each with coordinates p_x, p_y, p_z expressed with respect to the robot frame. The robot first computes a subset of candidate canopy points as follows

$$\mathcal{P}_{\text{canopy}} = \{\mathbf{p} \in \mathcal{P} \mid \text{atan2}(p_y, p_x) \in [\phi_{tree} \pm \Delta\phi] \wedge h_{\min} \leq p_z \leq h_{\max}\}$$

where ϕ_{tree} is the angle between the robot heading and the segment connecting $\mathbf{p}_{\text{robot}}$ with $\mathbf{p}_{\text{tree}}$, $\pm\Delta\phi$ is an error bound, and h_{min}, h_{max} are two values limiting the height range of the points to consider (set to 1.0m and 1.5m respectively in our experiments.) The goal point for the robot to navigate to is set as

$$\mathbf{p}_{\text{goal}} = \arg \min_{\mathbf{p} \in \mathcal{P}_{\text{canopy}}} ||\mathbf{p} - \mathbf{p}_{\text{robot}}|| - d\mathbf{n}_p$$

where $\mathbf{n}_p$ is the unit vector from $\mathbf{p}_{\text{robot}}$ to $\mathbf{p}$ and d is a safety distance (set to 1m in our experiments.) While not every leaf within $\pm\Delta\phi$ and height $[h_{min}, h_{max}]$ may be reachable by the robot, this gives the robot a good selection to present the Kinova arm's vision module for proximal sensing described in the next subsection.

Importantly, this approach does not require that the estimated GPS positions of the trees or the robot's current position be perfectly accurate. Since the algorithm sweeps a wide azimuth for the closest point to where the tree is expected to be from the estimated robot position, this implies that the robot will pick its target location based on the canopy shape as perceived by the 3D LiDAR. After $\mathbf{p}_{\text{robot}}$ has been determined, a velocity-based controller is used to reach it, rather than relying on GPS coordinates. While the EKF provides global pose estimates and GPS-based navigation is available, the lack of stable connectivity in outdoor orchards makes precise GPS navigation unreliable when navigating under large canopies commonly found in mature trees. Instead, the robot generates collision-aware velocity commands using ROS2's collision manager. Given the commands are purely velocity-based, the robot relies on the BT retry logic, which can capture task failures and retry an entire sequence. As an example, should the velocity command push the robot slightly off the directional vector required to face the tree, the leaf sampling task that follows might fail. Such failure will notify the underlying BT and will trigger a retry. While the described approach is relatively simple, as discussed in Section V, it enables robust navigation when tested in commercial orchards not preconditioned for robot operation. Moreover, its well defined software interface allows to easily replace it with a more sophisticated approach when necessary.

D. Leaf Target Perception and 3D Pose Estimation

To demonstrate the feasibility of the perception-to-manipulation cycle described above, we integrated an existing open-source RGB-D leaf detection and pose estimation pipeline as a supplementary module within our tree-specific autonomy stack [5]. In our system, this module is treated as a black box: given a registered RGB-D observation (I, D) where I is the RGB image and D is the depth information with intrinsic parameters $\mathbf{K}$, it outputs a list of leaf pose estimates in the camera frame. Concretely, as per [5], a pre-trained YOLOv8 instance segmentation model produces leaf masks, which are lifted to 3D by querying points from the RGB-D point cloud aligned with I, followed by basic filtering and outlier rejection to form per-leaf point sets and estimate centers. Candidate targets are transformed to the robot base frame using known extrinsics/TF and passed to MoveIt2 for motion planning; the manipulator evaluates

FIGURE 8. **The Kinova Kortex with a custom end effector tip on a Robotiq gripper grasping a citrus leaf during a lab experiment.**

FIGURE 9. **Amiga along side our potted citrus plant used in our lab-based experiments.**

FIGURE 10. **Our robot approaching a citrus tree.**

targets sequentially until a feasible collision-free trajectory is found, then closes the gripper and records a spectrum (see Figure 8.) We stress that in our architecture and experiments this block has been integrated "as is" to demonstrate the flexibility of the software stack we developed. Because our autonomy stack interfaces with this module only through its pose outputs, this component, too, can be readily replaced with alternative perception methods in future work.

V. Results

A. Experimental Setup

Before describing the results, we briefly describe the two different setups we utilized. The reader is also referred to the supplementary material for additional visuals showing the robot in action in different scenarios.

1) Lab Setup

Prior to evaluation in the field, we validate that the premise of this architecture works in a controlled environment. We designed this experiment to be conducted on flat terrain with good wheel traction to eliminate robot control issues, as are common in the field. Although the conditions are more ideal, we use a real potted citrus tree (tangerine) to demonstrate the robot's ability to approach and sample a leaf (see Figure 9). This step was necessary to fine-tune the parameters for the third-party leaf perception software [5] described above. While in field testing the robot always faces $\pm 90°$ relative to the intended tree to sample due to orchard-based navigation constraints, we envision a scenario in which sampling may occur from a variety of robot orientations. In this experiment, we demonstrate tree approaching and hyperspectral sampling from four different robot orientations with respect to the citrus tree: $0°$, $+90°$, $-90°$, and $180°$.

2) Field Setup

For our field experiments, we tested our system in two commercial orchards, one planted with pistachio trees and one planted with orange trees. While our robotic system is meant to be used in pistachio orchards, we use the citrus orchard as a gauge to demonstrate generalization of this system and where we might need to improve to scale this design.

To configure the mission planner, we provide a GPS polygon of the regions of interest along with the number of trees per row. In the pistachio orchard, we selected a 144-tree subsection (8 × 18) of the greater orchard. The exact overview is shown in Figure 6. In the citrus orchard, we selected a smaller 12-tree subsection (3 × 4) to avoid interfering with ongoing commercial farm operations. We aim to demonstrate the robot's field readiness through the following outdoor experiments:

- Approaching and sampling a single leaf in a controlled lab setting.
- Approaching and sampling a single leaf from various trees in orchard subplots;
- Approaching and sampling multiple trees in different tree rows requiring aisle-based navigation in the mission plan;
- Demonstrating this pipeline in both pistachio and citrus orchards.

3) Success Definition

We define *success* of a given trial by the ability to complete the mission and task execution, not by the data collected. If the navigation component sets the robot up for successful sensing pipeline execution, we claim success. For example, if a robot is told to visit several trees and grasp the leaves, this requires that the robot be at an appropriate distance from the desired trees and correctly oriented in order to launch the leaf segmentation and manipulator actions. If too far or wrongly oriented, the actions will not launch due to either no leaves detected or infeasible arm path planning. While ultimately we aim for this platform to be integrated into a variety of autonomous sensing pipelines, grading the quality of data collected is outside of the scope of this work, as we focus on navigation. Our intent is to demonstrate and assess the pipeline yielding autonomy in the orchard starting from NL mission specification. This requires a level of fault tolerance and precision to work successfully, regardless of the sensing use case.

B. Controlled Laboratory Validation

In all test cases, the lab control test successfully validated the design and integration of this system's components. From either an orthogonal or parallel orientation to the tree, the robot was able to identify the potted tree from a distance of between 1 and 5 meters. In each trial captured in Table 2, in two distinct lighting conditions, the robot was able to successfully approach the tree and launch an attempt to grasp and measure a leaf. We go into further detail of the reliability and value-add of this sensing in Subsection F.

C. Single-Tree Citrus Orchard Experiments

In this class of experiments, the robot is tasked with visiting only one tree within a row of other trees. More specifically, the robot was asked to sample five different trees in the orchard (see Figure 10) with the prompt, "*Sample leaves from tree number X*" where X is the desired tree number. From this request, the planner decomposes a set of tasks that send the robot to enter the correct row, identify the tree, isolate the closest leaf cluster, approach the tree, and launch an action to collect a sample using the hyperspectral gripper, with the results displayed in Table 3. In two of the missions, the robot was not initially positioned in such a way that the manipulator could identify and sample a leaf. By virtue of the fault tolerant BT, the robot repositioned itself and would correctly identify leaf to begin the sampling pipeline. The only failure mode was caused by a canopy structure that prevented accurate navigation and subsequent grasp planning; this case is discussed in Subsection G, where we address the limitations of our system.

TABLE 2. Single-tree lab experiments across multiple robot approach orientations.

Orientation	Trials	Recoveries	Success	Lighting
0°	5	2/2	5/5	
+90°	5	2/2	5/5	
−90°	5	1/1	5/5	Sunny
180°	5	3/3	5/5	
+90°	5	0/0	5/5	
−90°	5	0/0	5/5	Cloudy
Overall	**30**	8/8	30/30	

TABLE 3. Single mission, single-tree citrus tree experiment results.

Tree #	Trials	Success	Lighting
2	2	2/2	
3	1	0/1	
5	2	2/2	Dusk
7	1	1/1	
11	2	2/2	
Overall	**8**	7/8	

D. Multi-Row Orchard Experiments

Building upon the single-tree missions, we then evaluated the system's ability to perform sequential sampling across multiple trees in multiple rows. These experiments transition from single-tree interactions to continuous multi-row missions, reflecting the operational requirements of large-scale precision agriculture. In these trials, the robot was tasked with navigating to a sequence of requested trees, performing the approach and sampling maneuver at each one, and then proceeding to the next target without any

TABLE 4. Evaluation missions, LLM prompts, mission descriptions, average distance traveled and average mission duration. Distance traveled and mission duration are averaged over three trials. Mission Descriptions reflect test intent, not LLM input.

Mission	LLM Prompt	Mission Description	Avg. Dist.	Avg. Dur.
1	"Sample trees 11 and 13"	Sample two trees in two different rows	89.38 m	389.6 s
2	"Sample trees 11 and 14"	Sample two trees in two different rows, farther apart than Test 1	110.81 m	435.2 s
3	"Sample trees 11 and 53"	Sample two trees with one located deep inside the aisle	157.99 m	545.9 s
4	"Sample Trees 11, 13 and 14"	Sample three trees across three different rows	147.10 m	671.5 s
5	"Sample Trees 11, 13 and 29"	Sample three trees with two in the same row	136.72 m	576.3 s

manual intervention or additional NLP instructions (see also Figure 5.) The results are summarized in Tables 4 and 5.

1) Pistachio Orchard Experiments

In these experiments, using our proposed method the robot successfully navigated to and executed the sampling approach for 93.3% of the targeted pistachio trees across five multi-tree missions, each of which was repeated three times (see Table 5). Of the five different missions executed in these experiments, each was crafted with the intent to explore different navigational complexities. While the exact prompt for each set of trees was the same as in Subsection C, "*Sample leaves from tree numbers X, Y, and/or Z*", the decomposition of the tasks reflects slightly different complexity in each request. The list below qualitatively expresses the resulting complexity of each request stemming from a similar query. Samples of the paths followed by the robot in all five missions are shown in Figure 11.

- Mission 1: Approach two trees in adjacent rows
- Mission 2: Approach two trees, but skip a row in between
- Mission 3: Approach two trees in adjacent rows, but the second tree must be deeper into the row, requiring many intermediate waypoints to be added
- Mission 4: Approach three trees in three different rows
- Mission 5: Approach one tree in a row and two trees in an adjacent row

The lone failure case was an instance where the BT output from the LLM wasn't robust enough to correctly launch the leaf sampling action. In most cases, the LLM wrapped a retry loop around the approaching and sampling actions. However, at the discretion of the planner, the number of retries was defaulted to three. The failure occurred after three retries was not enough to properly sample. In four of the 15 trials, this retry loop was engaged with this one instance not being enough. An example of the produced retry loop is shown in Figure 12.

```
<RetryUntilSuccessful num_attempts="3">
  <Sequence>
    <MoveToTreeID name="approach_tree_10" id="10"
                  approach_tree="true"
                  action_name="follow_tree_id_waypoint"/>
    <SampleLeaf name="sample_tree_10"
                action_name="segment_leaves"/>
  </Sequence>
</RetryUntilSuccessful>
```

2) Ablation Study

Though Table 5 shows promising results for our method, we compare it to both the architecture that inspired this system, [17], and to an ablation study of critical features added to [17]. This comparison is meant to demonstrate that both key planning additions are essential for a robust, offline planning design. As described in Section IV, the key additions to the planner are leveraging BT.CPP as the default XML format and adding a module that makes LLM context more efficient, i.e., the OM module formerly described. Table 5 shows that while adding either of these features improves the base design, both features together are necessary to deploy the robot in a field setting. The improvements brought by the additions of both BT.CPP and OM showed in our experiment results very clearly.

First, without BT.CPP, the resulting mission was not fault tolerant. While there were only four mission failures without BT.CPP, the goal of our deployable system is autonomy. Failure of the proximal sensing occurred for roughly 25% of missions. Without a robust retry mechanism, a key part of BT.CPP and shown in Figure 12, the mission failed. This same principle can be applied to fallback logic, reactive sequences, and other core concepts of BTs. The most important takeaway being that the LLM planner is able to exploit the value of a robust BT specification and wraps critical tasks in fail-safes.

Lastly, without OM, the LLM planner is provided with every GPS points in the context to generate a mission. As substantiated also in [31], our system without this OM caused the planner to ignore system prompt instructions. The clearest example is when the planner instructed the robot to go to the western adjacent waypoint of a tree, sample the tree, and then go to the eastern waypoint of the same tree. While this might seem trivial, part of the system prompt, shown in Appendix B, explicitly requests that rows of trees not be traversed. Though the robot can fit through these trees, sometimes there are irrigation implements or branches that either the robot can damage or can cause damage to the robot. Since [31] calls out a 32k token context window as being a "cliff" with a max of 8k tokens as optimal, we show in Table 6 that our missions without OM are in the 10k+ range for an orchard of only 144 trees. This ablation showcases the same results found in [31] by the fact that there is not a single hallucination in the trials of our proposed solution.

TABLE 5. Mission Success Rate Comparison Across Test Tree Configurations from Table 4

	Pistachio Orchard Experiments (Success/Trials)					
Method	Mission 1	Mission 2	Mission 3	Mission 4	Mission 5	**Total**
Zuzuárregui & Carpin [17]	0/3	1/3	2/3	1/3	1/3	5/15
Ours, no BT.CPP	2/3	2/3	2/3	3/3	2/3	11/15
Ours, no OM	2/3	2/3	2/3	3/3	3/3	12/15
Ours, GPT-OSS 20B	0/3	0/3	0/3	0/3	0/3	0/15
Ours	3/3	2/3	3/3	3/3	3/3	**14/15**

(a) Mission 1 (b) Mission 2 (e) Mission 3

(c) Mission 4 (d) Mission 5

FIGURE 11. GPS path of pistachio orchard experiments carried out in Table 4.

FIGURE 12. A retry loop example from the XML planner output.

We further extended this ablation study to experimenting with local LLMs instead of the cloud based GPT 5.4 (third row in the table). We selected GPT-OSS-20B with "medium" reasoning as it is one of the most powerful edge compute models we can run with our hardware setup. The complexities of these missions, while seemingly trivial, were outside the capabilities of the local model. As noted in Table 5, not a single mission succeeded when planning locally. The reason for every single failure is due to the inability to decompose the mission properly and generate intermediate navigation tasks to navigate orchard aisles. Since many of these robots operate in environments without network connectivity, we wanted to explore if running a local model instead of a cloud-based one would suffice for our requirements. These results justify the architecture's design of requiring initial cloud-based planning prior to execution.

3) Tokens

In Subsection 2 we touched on context size being a critical factor for deployment of these LLM-based planning systems. We further investigate how each of the models in Table 5 deals with tokens. These results are shown in Table 6. As mentioned, input context size expands greatly without the OM module to compress the context by an algorithmic mapping of trees to GPS points and performance decreases. However, as the size of the orchard grows, our method's input context remains fixed. All other methods without context compression will grow linearly in size with each added tree. For example, our experiments in the pistachio orchard are on a subplot of 144 trees. This represents roughly a quarter of the total orchard. With commercial orchards being in the thousands of trees, we showed that a 2,000 tree orchard without our OM addition would account for a nearly 150k input token prompt and a roughly $0.40 per mission cost even though input tokens are a fraction of the cost of output tokens. As discussed in the previous subsection, local off-the-shelf models that can provide "free" prompting do

TABLE 6. Average token usage and cost across test tree configurations from Table 4. Cost is computed by OpenAI pricing of $2.50/1M input tokens and $15.00/1M output tokens, rounded up to the nearest cent.

	Orchard Experiments (Tokens In/Out, Cost $)									
Method	Mission 1		Mission 2		Mission 3		Mission 4		Mission 5	
[17]	13,328/2,919	$0.08	13,328/2,852	$0.08	13,328/3,055	$0.08	13,332/3,670	$0.09	13,332/3,308	$0.08
Ours, no BT	3,376/1,821	$0.04	3,376/777	$0.02	3,376/1,956	$0.04	3,379/1,499	$0.03	3,379/1,856	$0.04
Ours, no OM	13,464/3,034	$0.08	13,464/2,533	$0.07	13,464/3,260	$0.08	13,468/3,876	$0.09	13,468/4,198	$0.10
Ours, GPT-OSS	3,573/1,603	$0.00	3,573/3,131	$0.00	3,573/2,346	$0.00	3,577/1,132	$0.00	3,577/1,467	$0.00
Ours	3,512/1,290	$0.03	3,512/1,110	$0.03	3,512/863	$0.02	3,516/1,955	$0.04	3,516/1,462	$0.03

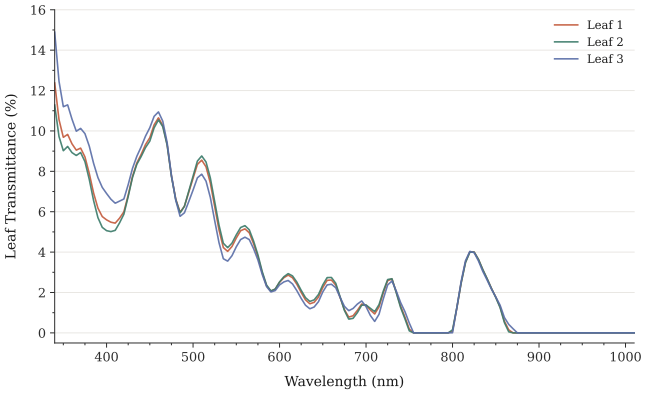


FIGURE 13. Hyperspectral signatures of 3 of the trials where leaf contact was successful.

not yet support the complexity required for these partially structured environments.

4) Does This Generalize?

We wanted to extend our experiments outside of the pistachio orchard to a small subplot of a commercial citrus orchard to investigate whether this system could generalize to an environment for which it wasn't build. In the citrus orchard, we evaluated the full sensing pipeline using multi-tree missions as was written in Table 4. We observed a couple of interesting findings not noted in our pistachio experiments. While the mission planner successfully guided the robot to the tree waypoints, the sensing action was sometimes unable to complete the leaf-grasping sequence. One of these failures was primarily caused by the low height and small size of the citrus trees, which allowed direct sunlight to reach the camera sensor and impair detection, as shown in Fig. 16, unlike the pistachio orchard where the taller and more mature trees blocked direct solar exposure. Additionally, as discussed in Subsection C, different canopy growth patterns prevented the robot from approaching the tree.

E. Long horizon Experiment

Building upon the multi-tree missions in subection V.D, we further evaluated the system by extending the planning horizon and modifying the NL mission query itself. Rather than enumerating specific tree identifiers, the robot was instructed with *"Sample all trees in the second aisle,"* a semantically broader command requiring the planner to resolve the full target set from context. Using our proposed architecture, the robot successfully sampled 17 of the 18 trees in the aisle, yielding a 94.4% success rate and demonstrating that the architecture scales gracefully to longer mission horizons without meaningful degradation in performance.

F. Hyperspectral Measurements and Proximal Sensing

The integration of a hyperspectral sensor enables the platform to serve as a comprehensive diagnostic tool for orchard health, providing a high-dimensional data source that extends far beyond water stress assessment. These spectral signatures allow for the identification of nutrient deficiencies [52], early-stage diseases [53], and pest monitoring [54] through markers that are often invisible to the naked eye. However, the primary focus of this work remains the MP and behavioral logic required to enable any kind of sensor to interact with a tree reliably and autonomously. In an orchard environment, the motion planning necessary to safely navigate a robotic arm into a complex canopy, while maintaining the stability required for a high-fidelity spectral reading, presents a significant challenge not covered in this work or in [5]. The leaf-grasping success rate was 60% over 60 grasp attempts, this is consistent with what we expected and with the performance reported by the author in [5]; grasp optimization was beyond our objectives. The system did successfully capture several high-quality spectral signatures during the single- and multi-tree missions. the system did successfully capture several high-quality spectral signatures during the single and multi-tree missions. Figure 13 illustrates a representative sample of the data collected, showing the reflectance curves of citrus leaves under varying conditions. This figure demonstrates that the platform can successfully execute the required leaf sensing trajectory planning required to produce scientifically relevant proximal

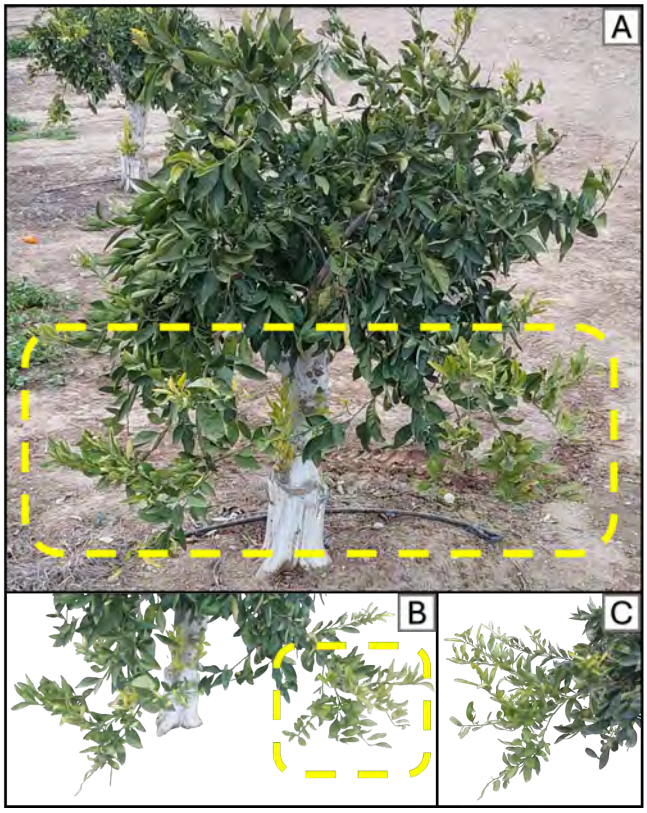


FIGURE 14. A) Branch shoots known as "suckers" due to rapid early spring growth. B) Area of interest causing robot approach problems. C) Side view of highlighted areas in A and B.

sensing data, effectively serving as a general-purpose vehicle for complex agricultural payloads.

By decoupling the sensing payload from the navigation and approach logic, we demonstrate that this platform is sensor-agnostic. The robot is capable of delivering specialized proximal sensing, whether hyperspectral, thermal, or chemical, to a precise biological target. This versatility transforms the robot from a single-purpose, human-driven tool into a general data collection framework. Researchers and orchard managers can swap the sensing module to address the specific seasonal needs of the orchard without altering the underlying BT or MP architecture. Consequently, the platform provides a scalable solution for proximal sensing in complex agricultural environments.

G. Limitations and Lessons Learned

While the proposed architecture successfully integrates mission planning with reactive BT-based sampling, several limitations were identified during field deployment that warrant further investigation and will guide our future work.

1) Lighting Sensitivity

Unsurprisingly, a primary bottleneck discovered in the citrus trials was the sensitivity of the vision-based sensing model

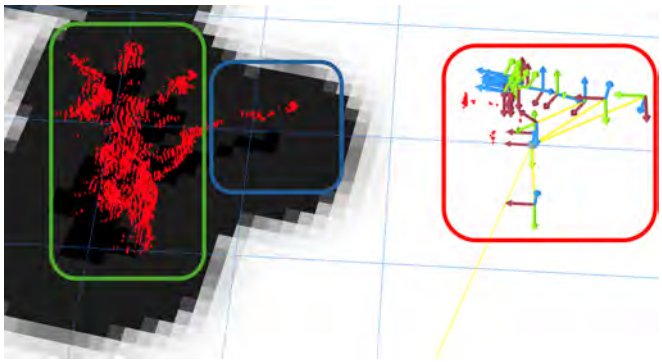


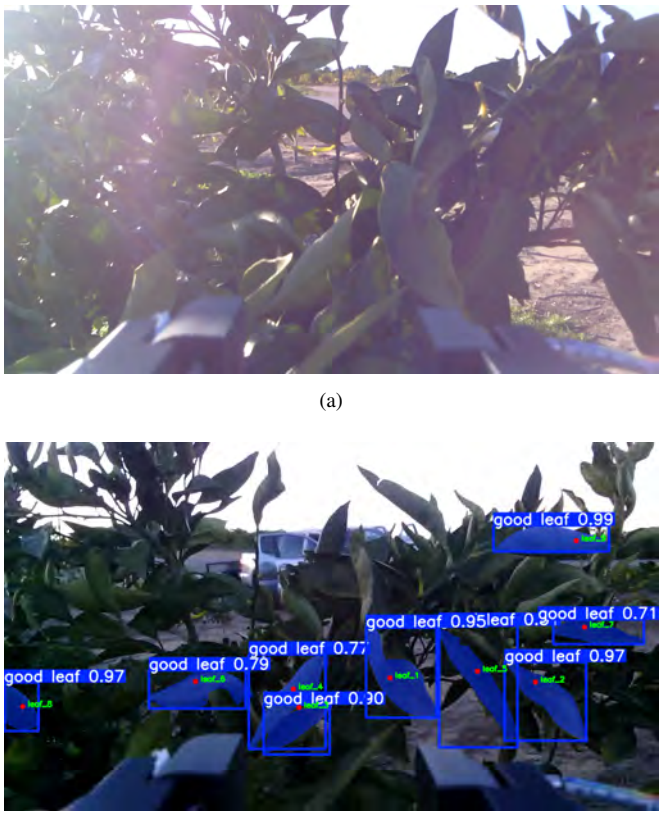
FIGURE 15. Costmap representation of what is shown in Figure 14. The green box is the body of the canopy. A portion of that is between h_{min} and h_{max}, eligible for sample selection. The blue box is a sucker branch sticking out from the tree. Since Nav2 is configured to avoid all collisions and this blue region is not within h_{min} and h_{max}, we see a stalemate where the robot cannot fulfill the request to reach $\mathrm{p}_{\mathrm{goal}}$ because the collision manager prevents the approach path. The red box shows the robot joint frames facing the tree

.

to direct sunlight. The model we adapted from [5], which performed well under indirect sunlight, struggled in scenarios that placed the arm directly facing the sun. Since we originally tested this vision segmentation model during pistachio season, we did not encounter such problems due to the height of fully mature pistachio trees. However, in the citrus orchard we tested, the trees we worked with were no more than two meters in height. While the robot may have had greater success in the adjacent citrus plot with mature, dense trees, our tests show that there is more variability to consider for fully autonomous data collection. It is, however, important to stress that, as previously outlined, our software architecture allows for the easy replacement of the software offered in [5] with any alternative that respects the same interface discussed in Section IV, and that leaf perception and pose estimation are not the focus of the work presented here.

2) Tree Canopy Shapes

Another limitation observed in the citrus trials was sensitivity to atypical canopy geometry. Our method estimates a safe stopping pose and sampling target by selecting the nearest feasible canopy region from the LiDAR point cloud. In trees with strongly non-uniform structure, this method can be biased by outlier geometry, such as a single elongated branch protruding well beyond the rest of the canopy. In these cases, the computed robot-to-canopy distance is dominated by the protruding branch rather than the main canopy surface, causing the robot to stop farther from the leaves than intended. Although the pose may be collision-safe, it can place the arm outside its effective reach, preventing successful sampling (see Figure 14 and 15 for one such case). In contrast, if the method reduces the height range of feasible canopy points under consideration, the robot could end up driving straight



(a)

(b)

FIGURE 16. Leaf segmentation output in the citrus orchard at solar noon with zero leaves detected (A) and later in the afternoon with 9 leaves detected (B).

into these low-hanging branches and potentially damaging them or itself.

Along the same theme, GPS-based navigation in areas with potential collisions has proven at times to be difficult for the ROS2 navigation stack. When relying on Nav2, the robot is subject to what the navigation stack deems a safe or unsafe path. Originally, the method used GPS-based navigation to place the robot directly in front of the tree for sampling. However, often times Nav2 was unable to generate a pose due to the parameterization governing how it can move using GPS with obstacle avoidance. Our next solution was to integrate velocity control, but this proved unsafe, as sometimes anomalistic LiDAR readings could cause collisions. Finally, we settled on a collision-aware velocity control node, but limitations still remain, as described in the paragraph above.

We also observed that canopy sparsity can affect navigation behavior between trees. Because trees are represented as obstacles using the front-camera point cloud, dense canopies typically yield conservative obstacle boundaries that discourage the robot from traversing gaps. In contrast, in younger or sparsely planted orchards, the perceived obstacles may not fully block the space between adjacent trees, and the robot may attempt to drive through the row interior when a collision-free path exists. While this can be kinematically feasible, it may be undesirable in practice due to unmodeled field hazards (e.g., sprinklers) that may be present between trees and could be damaged.

3) Leaf Grasping Variability

A final limitation is grasping variability due to leaf motion and leaf geometry. Under windy conditions, leaf clusters can move during the approach, reducing the repeatability of the final contact pose. In addition, citrus leaves are often small, densely packed, and not perfectly planar; slight curvature can cause the gripper fingers to slide or push the leaf away during closure. These factors can lead to partial grasps (e.g., contacting only the leaf tip) or inadvertently grasping multiple leaves. In such cases, the optical sensor may not fully interface with a single leaf surface, which can degrade measurement quality and reduce the consistency of the recorded spectra. As stated above, this may be mitigated by switching to a different software package for leaf detection.

VI. Discussion

Navigation in environments with tight clearances and potential collisions remains challenging. When relying on Nav2, the robot is constrained by the costmaps and local planner, which determine whether a candidate trajectory is considered safe or unsafe. In early iterations, we used GPS-based navigation to command the robot to a pose directly in front of the target tree. In practice, Nav2 frequently failed to generate a valid goal or trajectory near the canopy, largely due to the parameterization of the GPS navigation behavior and the motion constraints enforced by the local planner. This highlights that Nav2 performance is highly sensitive to application-specific tuning, including costmap inflation, obstacle layers, footprint modeling, and controller parameters; achieving reliable tree-proximal behavior required careful calibration for the orchard geometry and desired approach distances.

Field conditions further motivate reducing reliance on GPS alone. In orchard environments, GPS quality can degrade or be lost under canopy, and this degradation can occur precisely when the robot approaches the tree for sampling. As a result, purely GPS-driven control can become unreliable during the final approach. A more robust strategy is to fuse GPS with a high-quality, calibrated IMU (and wheel odometry, when available) using an EKF, allowing the system to maintain stable state estimates during intermittent GNSS outages and enabling consistent close-range navigation and manipulation.

VII. Conclusions and Future Work

This paper presented an end-to-end software architecture designed to bridge the gap between high-level NL MP and low-level reactive control for autonomous robots in precision agriculture. By leveraging LLMs, our system successfully

translates user intent into structured XML specifications, which are then parsed into fault-tolerant BTs for execution. To address the limitations of static waypoint databases and LLM context constraints, we introduced an orchard management abstraction layer that dynamically interpolates navigation targets from user-defined GPS polygons.

Through the use of standard components of ROS2 (e.g., Nav2 and MoveIt2) and multi-modal perception modules, the architecture decoupled mission orchestration from hardware-specific execution. Our experimental validations, conducted both in controlled laboratory settings and commercial pistachio and citrus orchards, demonstrated the system's capability to navigate semi-unstructured environments and reliably approach target canopies for proximal leaf sensing. The retry logic embedded in the BT execution proved critical to recover from runtime failures.

Future work will focus on enhancing the perception pipeline to be robust against dynamic lighting and wind conditions, as well as refining the collision-aware approach logic for irregular canopies.

Appendix A
XSD Schema Action Pool

We provide a snipped of the XML schema passed as context to the LLM planner. The complete XML schema can be found the the following anonymized website: https://anonymous.4open.science/r/gpt-mission-planner-schemas-B70D/amiga_btcpp.xsd

```
<!-- Robot actions -->
<xs:group name="ActionGroup">
    <xs:choice>
        <xs:element name="MoveToTreeID"
            type="moveToTreeIDType" />
        <xs:element name="MoveToAisleHead"
            type="moveToAisleHeadType" />
        <xs:element name="MoveToGPSLocation"
            type="moveToGPSLocationType" />
        <xs:element name="MoveToRelativeLocation"
            type="moveToRelativeLocationType" />
        <xs:element name="OrientRobotHeading"
            type="orientRobotHeadingType" />
        <xs:element name="FollowPerson"
            type="followPersonType" />
        <xs:element name="MoveArmToPosition"
            type="moveArmToPositionType" />
        <xs:element name="SampleLeaf"
            type="sampleLeafType" />
    </xs:choice>
</xs:group>
```

Appendix B
System Prompt

Below is the system prompt that defines how the mission generation agent acts. It sets several syntactic rules and defines basic principles of BT.CPP. Finally, it injects the schema of choice, based on which robot is being used, and the associated context files described in Appendix C. The following uses Jinja2 for text injection based on user configuration in the frontend webapp.

```
You are an assistant that interprets and generates
    behavior tree missions in XML.

Important semantic rules (MANDATORY):

1. Output requirements
All responses MUST be valid XML that conforms to the
    provided XSD and reflects correct behavior tree
    semantics.
You MUST include the original mission request verbatim
     in the Mission element.
Respond with a single XML markdown code block only, in
     the form:
```xml
<root>...</root>

2. Do NOT include comments or XML declarations.

3. Fallback or selector nodes MUST be used ONLY when
    the mission explicitly specifies multiple
    distinct strategies or when multiple semantically
     different actions exist to achieve the same goal
    .
Fallback nodes MUST NOT be used to model retries,
    recovery, or alternative parameterizations of the
     same strategy.

4. Only actions that are critical to mission success--
    such as approaching a target tree (approach_tree
    ="true") or performing a sampling/manipulation
    action--must be placed inside a retry control
    structure.
Note that they should be placed together if they are
    critical to the success of the same subgoal.
Transit, intermediate moves, or relative moves that
    merely navigate between waypoints must not be
    retried.


{% include "schemas.j2" %}
{% include "context_files.j2" %}
```

Appendix C
Additional Context

This context is injected with user input populated at mission generation time. The user must select the appropriate farm to work on, which comes with a GPS polygon and orchard dimensions. However, this information can be easily updated via YAML files should a new farm be requested. The following uses Jinja2 for text injection based on user configuration in the frontend webapp.

```
{% set rows = farm_polygon.dimensions[0].row %}
{% set cols = farm_polygon.dimensions[0].col %}
{% set top_left = farm_polygon.points[0] %}
{% set top_right = farm_polygon.points[1] %}
{% set bottom_right = farm_polygon.points[2] %}
{% set bottom_left = farm_polygon.points[3] %}
{% set top_right_tree = cols %}
{% set bottom_left_tree = (rows - 1) * cols + 1 %}
{% set bottom_right_tree = rows * cols %}
FARM LAYOUT:
Top Left (NW) (tree 1):
    {{ top_left.lat }}, {{ top_left.lon }}
Top Right (NE) (tree {{ top_right_tree }}):
    {{ top_right.lat }}, {{ top_right.lon }}
Bottom Left (SW) (tree {{ bottom_left_tree }}):
    {{ bottom_left.lat }}, {{ bottom_left.lon }}
Bottom Right (SE) (tree {{ bottom_right_tree }}):
    {{ bottom_right.lat }}, {{ bottom_right.lon }}
Grid size:
{{ cols }} trees per row
```
```

```
{{ rows }} trees per columns
Total trees:
    {{ cols }} x {{ rows }} = {{ rows * cols }} trees
Traversal axis:
    {{ farm_polygon.traversal_axis }}
RULES:
- Numbering: left-to-right, top-to-bottom
- You CANNOT drive through {{
    farm_polygon.traversal_axis }}s of trees
- You MUST move to {{ farm_polygon.traversal_axis }}
    head when entering and exiting
    {{ farm_polygon.traversal_axis }}s.
- If traversal between trees is long, plan a path that
    includes intermediate waypoints.
USE THIS INFORMATION TO CREATE THE XML MISSION.
```

Additionally, we provide a small set of rules for the management of the Kinova Kortex movement. This allows the LLM planner to understand transform frame relationships, something LLMs appear to struggle with, using a well known standard.

```
Kinova Kortex Manipulator Orientation Rules

Use REP-103 names:
Roll = rotate about X
Pitch = rotate about Y
Yaw = rotate about Z

BUT apply them in the end-effector's own coordinate
    frame, defined as:
+X = left
+Y = up
+Z = forward (tool pointing direction)

Because the tool points along +Z instead of REP-103's
    usual +X, the meanings of the rotations shift:

Resulting Motion Meanings (with signs)
1. Turn left / right -> Pitch (+/- about Y)
Rotation about the Y axis rotates the forward axis
    (+Z) toward left or right.
+pitch = turn left (rotate +Z toward +X)
-pitch = turn right (rotate +Z toward -X)

2. Tilt up / down -> Roll (+/- about X)
Rotation about the X axis rotates the forward axis
    (+Z) toward up or down.
+roll = tilt down (rotate +Z toward -Y)
-roll = tilt up (rotate +Z toward +Y)

3. Twist clockwise / counter-clockwise -> Yaw (+/-
     about Z)
Rotation about the Z axis spins around the tool's own
    forward direction.
+yaw = twist clockwise (right-hand rule)
-yaw = twist counter-clockwise
```

D ROS2 Parameters

We finetuned Nav2 to better suit our robots needs when navigating within the orchard. Below are the Nav2 specific parameters changes.

```
bt_navigator:
  ros__parameters:
    odom_topic: /odometry/filtered/global
    default_server_timeout: 60

controller_server:
  ros__parameters:
    odom_topic: /odometry/filtered/local
    goal_checker:
      yaw_goal_tolerance: 3.14
    FollowPath:
      min_lookahead_dist: 0.8
      max_lookahead_dist: 2.5
      use_velocity_scaled_lookahead_dist: true
      use_rotate_to_heading: true
      approach_velocity_scaling_dist: 1.0

local_costmap:
  local_costmap:
    ros__parameters:
      publish_frequency: 2.0
      width: 8
      height: 8
      footprint: "[ [-0.57, 0.7], [0.57, 0.7], [0.57,
          -0.7], [-0.57, -0.7] ]"
      plugins: ["obstacle_layer", "inflation_layer"]
      obstacle_layer:
        observation_sources: oak0_pointcloud
      inflation_layer:
        inflation_radius: 2.0

global_costmap:
  global_costmap:
    ros__parameters:
      update_frequency: 2.0
      publish_frequency: 1.0
      rolling_window: true
      width: 75
      height: 75
      footprint: "[ [-0.57, 0.7], [0.57, 0.7], [0.57,
          -0.7], [-0.57, -0.7] ]"
      plugins: ["obstacle_layer", "inflation_layer"]
      obstacle_layer:
        observation_sources: oak0_pointcloud
      inflation_layer:
        inflation_radius: 2.0

planner_server:
  ros__parameters:
    GridBased:
      allow_unknown: true
```

We also added a collision monitor, not native to Nav2 in ROS2 Humble, for collision avoidant velocity control. Below are the parameters.

```
collision_monitor:
  ros__parameters:
    enabled: True
    base_frame_id: "base_link"
    odom_frame_id: "odom"
    cmd_vel_in_topic: "cmd_vel_raw"
    cmd_vel_out_topic: "cmd_vel"
    state_topic: "collision_monitor_state"
    transform_tolerance: 2.0
    source_timeout: 2.0
    stop_pub_timeout: 0.1
    observation_sources: ["oak0_pointcloud"]
    oak0_pointcloud:
      topic: "/oak0/points"
      type: "pointcloud"
      expected_update_rate: 1.0
      min_obstacle_height: 0.5
      max_obstacle_height: 1.5
      marking: true
      clearing: true
      obstacle_max_range: 5.0
      obstacle_min_range: 1.5
      raytrace_max_range: 5.0
      raytrace_min_range: 0.0
    polygons: ["PolygonStop", "ApproachFootprint"]
    PolygonStop:
      type: "polygon"
```

```
    points: [0.58, 0.7, 0.58, -0.7, -0.58, -0.7,
        -0.58, 0.7]
    action_type: "stop"
    max_points: 250
    visualize: True
  ApproachFootprint:
    type: "polygon"
    points: [0.85, 0.8, 0.85, -0.8, -0.85, -0.8,
        -0.85, 0.8]
    action_type: "approach"
    time_before_collision: 0.5
    max_points: 250
    visualize: True
```

For the Kinova Kortex, we used the default parameter set provided by the vendor for MoveIt2. The only change we made was in which planner the arm used for inverse kinematics. We opted to use PILZ planner instead of the default OMPL.

REFERENCES

[1] A. Dutta, S. Roy, O. P. Kreidl, and L. Bölöni, "Multi-robot information gathering for precision agriculture: Current state, scope, and challenges," *IEEE Access*, vol. 9, pp. 161 416–161 430, 2021.

[2] R. Singh, R. Berkvens, and M. Weyn, "Agrifusion: An architecture for iot and emerging technologies based on a precision agriculture survey," *IEEE Access*, vol. 9, pp. 136 253–136 283, 2021.

[3] J. Cuaran, K. Koe, A. Potnis, N. Uppalapati, and G. Chowdhary, "Visual-language-guided task planning for horticultural robots," 2026. [Online]. Available: https://arxiv.org/abs/2601.11906

[4] G. Mon-Williams, R.and Li, R. Long, W. Du, and C. Lucas, "Embodied large language models enable robots to complete complex tasks in unpredictable environments," *Nature Machine Intelligence*, vol. 7, no. 4, pp. 592–601, 2025.

[5] M. Mortazavi, D. J. Cappelleri, and R. Ehsani, "Romu4o: A robotic manipulation unit for orchard operations automating proximal hyperspectral leaf sensing," *arXiv preprint arXiv:2501.10621*, 2025.

[6] H. Kang, H. Zhou, X. Wang, and C. Chen, "Real-time fruit recognition and grasping estimation for robotic apple harvesting," *Sensors*, vol. 20, no. 19, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/19/5670

[7] B. Suthakar, A. Surendrakumar, R. Kavitha, P. Masilamani, *et al.*, "Tree fruit harvesting: Recent developments and future challenges for robotic harvesting," *Plant Science Today*, vol. 12, p. 8239, 2025.

[8] I.-S. Oh and J.-S. Lee, "A crawling review of fruit tree image segmentation," *Agriculture*, vol. 15, no. 21, 2025. [Online]. Available: https://www.mdpi.com/2077-0472/15/21/2239

[9] Z. Chen, X. Lei, Q. Yuan, Y. Qi, Z. Ma, S. Qian, and X. Lyu, "Key technologies for autonomous fruit- and vegetable-picking robots: A review," *Agronomy*, vol. 14, no. 10, 2024. [Online]. Available: https://www.mdpi.com/2073-4395/14/10/2233

[10] J. Dukić, P. Pejić, I. Vidović, and E. K. Nyarko, "Towards robotic pruning: Automated annotation and prediction of branches for pruning on trees reconstructed using rgb-d images," *Sensors*, vol. 25, no. 18, 2025. [Online]. Available: https://www.mdpi.com/1424-8220/25/18/5648

[11] A. Navone, M. Martini, and M. Chiaberge, "Autonomous robotic pruning in orchards and vineyards: A review," *Smart Agricultural Technology*, vol. 12, p. 101283, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772375525005143

[12] A. Carella, P. T. Bulacio Fischer, R. Massenti, and R. Lo Bianco, "Continuous plant-based and remote sensing for determination of fruit tree water status," *Horticulturae*, vol. 10, no. 5, 2024. [Online]. Available: https://www.mdpi.com/2311-7524/10/5/516

[13] 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. [Online]. Available: https://ntrs.nasa.gov/citations/19940019393

[14] S. Kalluraya *et al.*, "Multi-robot Mission Planning in Dynamic Semantic Environments," Mar. 2023, arXiv:2209.06323 [cs, eess].

[15] M. Askarpour, C. Menghi, G. Belli, 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*, ser. FormaliSE '20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 55–65. [Online]. Available: https://doi.org/10.1145/3372020.3391561

[16] B. Brumitt and A. Stentz, "Dynamic mission planning for multiple mobile robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, 1996, pp. 2396–2401 vol.3. [Online]. Available: https://ieeexplore.ieee.org/document/506522/?arnumber=506522

[17] M. A. Zuzuárregui and S. Carpin, "Leveraging LLMs for mission planning in precision agriculture," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2025, pp. 7146–7152.

[18] S. Kannan *et al.*, "SMART-LLM: Smart Multi-Agent Robot Task Planning using Large Language Models," Mar. 2024, arXiv:2309.10062 [cs].

[19] C. Mower *et al.*, "ROS-LLM: A ROS framework for embodied AI with task feedback and structured reasoning," July 2024, arXiv:2406.19741 [cs].

[20] J. Liang *et al.*, "Code as Policies: Language Model Programs for Embodied Control," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2023, pp. 9493–9500.

[21] W. Huang *et al.*, "Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents," Mar. 2022, arXiv:2201.07207 [cs].

[22] R. Emsley, "ChatGPT: these are not hallucinations – they're fabrications and falsifications," *Schizophrenia*, vol. 9, no. 1, pp. 1–2, Aug. 2023, publisher: Nature Publishing Group.

[23] M. Ahn *et al.*, "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances," Aug. 2022, arXiv:2204.01691 [cs].

[24] J. X. Liu, Z. Yang, I. Idrees, S. Liang, B. Schornstein, S. Tellex, and A. Shah, "Grounding complex natural language commands for temporal tasks in unseen environments," in *Proceedings of The 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 06–09 Nov 2023, pp. 1084–1110. [Online]. Available: https://proceedings.mlr.press/v229/liu23d.html

[25] J. Cuaran, K. Koe, A. Potnis, N. K. Uppalapati, and G. Chowdhary, "Visual-language-guided task planning for horticultural robots," 2026. [Online]. Available: https://arxiv.org/abs/2601.11906

[26] P. Gupta, D. Isele, E. Sachdeva, P.-H. Huang, B. Dariush, K. Lee, and S. Bae, "Generalized mission planning for heterogeneous multi-robot teams via llm-constructed hierarchical trees," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 10 187–10 193.

[27] M. A. Zuzuárregui, M. Toslak, and S. Carpin, "One for all: Llm-based heterogeneous mission planning in precision agriculture," in *Proceedings of the IFAC Conference on Sensing, Control and Automation Technologies for Agriculture*, 2025.

[28] M. A. Zuzuárregui and S. Carpin, "As you wish: Mission planning with formal verification using llms in precision agriculture." in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2026.

[29] J. Ao *et al.*, "LLM-as-BT-Planner: Leveraging LLMs for Behavior Tree Generation in Robot Task Planning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, May 2025, pp. 1233–1239. [Online]. Available: https://ieeexplore.ieee.org/document/11128454

[30] J. Styrud, M. Iovino, M. Norrlöf, M. Björkman, and C. Smith, "Automatic behavior tree expansion with llms for robotic manipulation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 1225–1232.

[31] A. Modarressi, H. Deilamsalehy, F. Dernoncourt, T. Bui, R. A. Rossi, S. Yoon, and H. Schütze, "NoLiMa: Long-Context Evaluation Beyond Literal Matching," Feb. 2025, arXiv:2502.05167 [cs.CL] version: 1. [Online]. Available: http://arxiv.org/abs/2502.05167

[32] J. Chen, H. Qiang, J. Wu, G. Xu, Z. Wang, and X. Liu, "Extracting the navigation path of atomato-cucumbergreenhouse robot based on a median point hough transform," *Computers and Electronics in Agriculture*, vol. 174, p. 105472, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919324172

[33] J. Yu, J. Zhang, A. Shu, Y. Chen, J. Chen, Y. Yang, W. Tang, and Y. Zhang, "Study of convolutional neural network-based semantic segmentation methods on edge intelligence devices for

field agricultural robot navigation line extraction," *Computers and Electronics in Agriculture*, vol. 209, p. 107811, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169923001990

[34] X. Li, J. Su, Z. Yue, and F. Duan, "Adaptive multi-roi agricultural robot navigation line extraction based on image semantic segmentation," *Sensors*, vol. 22, no. 20, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/20/7707

[35] R. Galati, G. Mantriota, and G. Reina, "Robonav: An affordable yet highly accurate navigation system for autonomous agricultural robots," *Robotics*, vol. 11, no. 5, 2022. [Online]. Available: https://www.mdpi.com/2218-6581/11/5/99

[36] R. Xu and C. Li, "A modular agricultural robotic system (mars) for precision farming: Concept and implementation," *Journal of Field Robotics*, vol. 39, no. 4, pp. 387–409, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22056

[37] S. Jiang, P. Qi, L. Han, L. Liu, Y. Li, Z. Huang, Y. Liu, and X. He, "Navigation system for orchard spraying robot based on 3d lidar slam with ndt_icp point cloud registration," *Computers and Electronics in Agriculture*, vol. 220, p. 108870, 2024.

[38] Z. Su, W. Zou, C. Zhai, H. Tan, S. Yang, and X. Qin, "Design of an autonomous orchard navigation system based on multi-sensor fusion," *Agronomy*, vol. 14, no. 12, 2024. [Online]. Available: https://www.mdpi.com/2073-4395/14/12/2825

[39] M. V. Gasparino, V. A. Higuti, A. N. Sivakumar, A. E. Velasquez, M. Becker, and G. Chowdhary, "Cropnav: a framework for autonomous navigation in real farms," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 824–11 830.

[40] Z. Li, R. Xu, C. Li, and L. Fu, "Visual navigation and crop mapping of a phenotyping robot mars-phenobot in simulation," *Smart Agricultural Technology*, vol. 11, p. 100910, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772375525001431

[41] C.-T. Vu, H.-C. Chen, and Y.-C. Liu, "Toward autonomous navigation for agriculture robots in orchard farming," in *2024 IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE)*, 2024, pp. 1–8.

[42] K. Kim, A. Deb, and D. J. Cappelleri, "P-agbot: In-row & under-canopy agricultural robot for monitoring and physical sampling," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7942–7949, 2022.

[43] ——, "P-agslam: In-row and under-canopy slam for agricultural monitoring in cornfields," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 4982–4989, 2024.

[44] ——, "P-agnav: Range view-based autonomous navigation system for cornfields," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 3366–3373, 2025.

[45] R. Cordova-Cardenas, L. Emmi, and P. Gonzalez-de Santos, "Enabling autonomous navigation on the farm: A mission planner for agricultural tasks," *Agriculture*, vol. 13, no. 12, 2023. [Online]. Available: https://www.mdpi.com/2077-0472/13/12/2181

[46] J. Kang, Z. Zhang, Y. Liang, and X. Chen, "Autonomous navigation of agricultural robots utilizing path points and deep reinforcement learning," in *2024 International Joint Conference on Neural Networks (IJCNN)*, 2024, pp. 1–8.

[47] L. Cui, F. Le, X. Xue, T. Sun, and Y. Jiao, "Design and experiment of an agricultural field management robot and its navigation control system," *Agronomy*, vol. 14, no. 4, 2024. [Online]. Available: https://www.mdpi.com/2073-4395/14/4/654

[48] H. Suk, D. Lee, M. Kim, S. Lim, and M. Won, "Orchard navigation algorithm for self-driving vehicles under poor gps signal," *Journal of Electrical Engineering & Technology*, vol. 20, no. 4, pp. 2747–2762, 2025.

[49] S. Cerrato, V. Mazzia, F. Salvetti, M. Martini, S. Angarano, A. Navone, and M. Chiaberge, "A deep learning driven algorithmic pipeline for autonomous navigation in row-based crops," *IEEE Access*, vol. 12, pp. 138 306–138 318, 2024.

[50] D. Faconti, "BehaviorTree.CPP — Behavior Trees Library in C++," https://github.com/BehaviorTree/BehaviorTree.CPP, accessed: 2025-10-23.

[51] IEEE, "IEEE Standard for Robot Task Representation," *IEEE Std 1872.1-2024*, pp. 1–32, 2024.

[52] H. D. D. Nguyen, V. Pan, C. Pham, R. Valdez, K. Doan, and C. Nansen, "Night-based hyperspectral imaging to study association of horticultural crop leaf reflectance and nutrient status," *Computers and Electronics in Agriculture*, vol. 173, p. 105458, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919309287

[53] S. Sankaran, A. Mishra, R. Ehsani, and C. Davis, "A review of advanced techniques for detecting plant diseases," *Computers and Electronics in Agriculture*, vol. 72, no. 1, pp. 1–13, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169910000438

[54] J. Zhang, Y. Huang, R. Pu, P. Gonzalez-Moreno, L. Yuan, K. Wu, and W. Huang, "Monitoring plant diseases and pests through remote sensing technology: A review," *Computers and Electronics in Agriculture*, vol. 165, p. 104943, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016816991930290X