**Project Documentation: ROS2-Based Agricultural Rover Localization and Navigation Simulation**

**1. Introduction**

This project involved building a comprehensive ROS2-based simulation environment for autonomous navigation in farmland conditions. Leveraging the capabilities of ROS2 Humble and Gazebo, the focus was on creating a photorealistic maize field, equipping a Husky rover with GNSS and monocular camera sensors, and implementing sensor fusion for real-time localization. The goal was to simulate row-following, validate pose estimation methods, and lay the foundation for scalable precision farming systems.

**2. Objectives**

* Simulate a 3D agricultural environment with accurate terrain, crop models, and lighting
* Configure a mobile rover (Husky) with GNSS, IMU, and monocular visual odometry (VO) sensors
* Build and debug a GNSS plugin in ROS2 and validate the published /fix topic
* Implement monocular VO using OpenCV in Python and align estimated trajectories with camera info
* Fuse GNSS and VO data using an EKF node in the robot\_localization package
* Visualize real-time rover motion using RViz2 and evaluate localization drift/smoothness

**3. Simulation Environment Setup**

**Farmland World Design**

* Used the virtual\_maize\_field ROS2 package to procedurally generate maize crop layouts
* Manually edited the .world file to create dense maize rows by duplicating and rotating models for visual realism
* Replaced base terrain texture with a high-resolution green field image using Blender, troubleshooting transparency and UV mapping issues

**Husky Rover Configuration**

* Integrated the Clearpath Husky SDF and URDF models into the Gazebo simulation
* Added a monocular front-facing camera and a GNSS receiver to the robot’s sensor stack
* Ensured compatibility with ROS2 topics /front\_camera/image\_raw and /fix for downstream processing

**4. GNSS Sensor Integration and Debugging**

**Plugin Integration**

* Identified missing libgazebo\_ros\_gps.so plugin from standard ROS2 Humble distribution
* Cloned and built the gazebo\_ros\_pkgs repository from source to compile the plugin
* Configured the GNSS <sensor> block in the robot’s SDF with appropriate Gaussian noise parameters and frame alignment

**GNSS Behavior Validation**

* Verified GNSS data publication on /fix with proper coordinate frames and update rates
* Defined <spherical\_coordinates> in the world file to simulate real geodetic coordinates
* Visualized GNSS paths in RViz2 and compared them with expected rover movement

**5. Monocular Visual Odometry**

**VO Implementation**

* Developed a custom ROS2 node in Python using OpenCV and cv\_bridge
* Extracted ORB features, performed feature matching, and estimated pose using the Essential matrix
* Recovered translation vectors and normalized them to apply to the robot’s trajectory

**Debugging and Refinement**

* Retrieved correct camera intrinsics (fx, fy, cx, cy) from /camera\_info and validated alignment
* Added RANSAC-based inlier filtering to reject outlier feature matches
* Introduced planar motion constraint by zeroing out vertical motion (z = 0) for rover-level navigation
* Validated stability by logging estimated trajectory and comparing with GNSS output

**6. Sensor Fusion Using EKF**

**Robot Localization Pipeline**

* Installed and configured robot\_localization to fuse GNSS and VO using ekf\_filter\_node
* Used navsat\_transform\_node to convert /fix (NavSatFix) to UTM coordinates
* Applied static transforms between base\_link, odom, and gps\_link to ensure consistent frame alignment

**EKF Node Setup**

* Created two EKF configurations: one with VO + IMU, and another with GNSS + IMU
* Tuned covariance matrices for X/Y localization accuracy and adjusted based on GNSS reliability
* Launched fusion stack and monitored /odometry/filtered and /odometry/gps in RViz2

**7. Visualization and Benchmarking**

* Launched RViz2 with custom configuration to display all relevant frames: /base\_link, /odom, /map, /fix, /camera\_link
* Logged trajectories for /fix, /odometry/vo, and /odometry/filtered during row-following simulations
* Compared each path to identify drift reduction and heading stability post-fusion
* Ran simulations in two test scenarios: open field (ideal GNSS) and partial canopy (degraded GNSS)

**8. Challenges Encountered**

* Plugin installation failures due to missing GNSS binary; resolved with manual source build
* VO instability from mismatched camera intrinsics and unfiltered feature matches
* Frame misalignment between GNSS and robot base; corrected with static transform setup
* Transparency issues in crop models due to Blender-to-Gazebo material incompatibility

**9. Key Outcomes**

* Fully functional GNSS-VO fusion pipeline running in ROS2 and Gazebo
* Accurate simulated farmland with visual crop rows, enabling navigation testing
* Realistic rover behavior under fused localization, usable for path planning and control integration
* Reusable ROS2 launch files and parameter configurations for modular development

**10. Future Work**

* Integrate ORB-SLAM3 or VINS-Fusion for advanced visual odometry
* Add 3D LiDAR and perform multi-sensor fusion including point cloud alignment
* Extend to multi-robot coordination for swarm-based farming simulation
* Apply reinforcement learning for path control under perception uncertainty

**Author**: Adharsh Prasad Natesan  
**Project Duration**: Spring 2025  
**Tools**: ROS2 Humble, Gazebo, OpenCV, RViz2, Python  
**Repository**: [Add GitHub Link Here]