InternPro Weekly Progress Update

Name	Email	Project Name	NDA/ Non-NDA	InternPro Start Date	OPT
Adharsh Prasad Natesan	anatesan@asu.edu	IT-Core Foundation Suriname	Non-NDA	2024-08-05	Yes

Progress

Include an itemized list of the tasks you completed this week.

#	Action Item/ Explanation	Total Time This Week (hours)
1	Creating a GNSS sensor for the husky	3
2	Debugging the GNSS sensor	3
3	Debugging and Refining Visual Odometry for Monocular Camera Input	3
4	GNSS-based Localization for Autonomous Vehicles: Prospects and Challenges	3
5	ORB-SLAM: a Versatile and Accurate Monocular SLAM System	3
6	M2C-GVIO: Motion Manifold Constraint Aided GNSS-Visual-Inertial Odometry for Ground Vehicles	3
7	GNSS-VO Integration and Validation	1
8	Report Writing	1
	Total hours for the week:	20

Verification Documentation:

Action Item 1: Creating a GNSS sensor for the husky – 3 hour(s).

Project Work Summary

- I created a GNSS module for the simulated Husky robot to generate GPS data compatible with ROS2. I began by identifying the required ROS-Gazebo plugin, libgazebo_ros_gps.so, which is responsible for publishing GNSS data to the /fix topic.
- Initially, this plugin was not available in my ROS2 Humble installation, so I attempted a binary install using ros-humble-gazebo-ros-pkgs and ros-humble-gps-tools. However, the plugin remained missing, so I manually built the gazebo_ros_pkgs repository from source, targeting the humble branch. This ensured the GPS plugin was compiled and correctly registered.
- After confirming the plugin was available, I updated the Husky robot's SDF file by creating a new gps_link and adding a <sensor type="gps"> block to it. This included a fixed joint to base_link, a pose offset to position the GNSS antenna above the chassis, and a Gaussian noise model for both horizontal and vertical accuracy.
- I also included the gazebo_ros_gps plugin inside the sensor definition, remapping the output topic to /fix and assigning a frame_id of gps_link.
- Lastly, I modified the simulation world file by adding a <spherical_coordinates> block to define the real-world geodetic reference point for the GNSS readings. This included latitude, longitude, elevation, and heading based on a real location.
- With these changes, the Husky robot now publishes GNSS data as sensor_msgs/NavSatFix at a configurable update rate, forming the basis for future sensor fusion with visual odometry.

Action Item 2: Debugging the GNSS sensor – 3 hour(s).

Project Work Summary

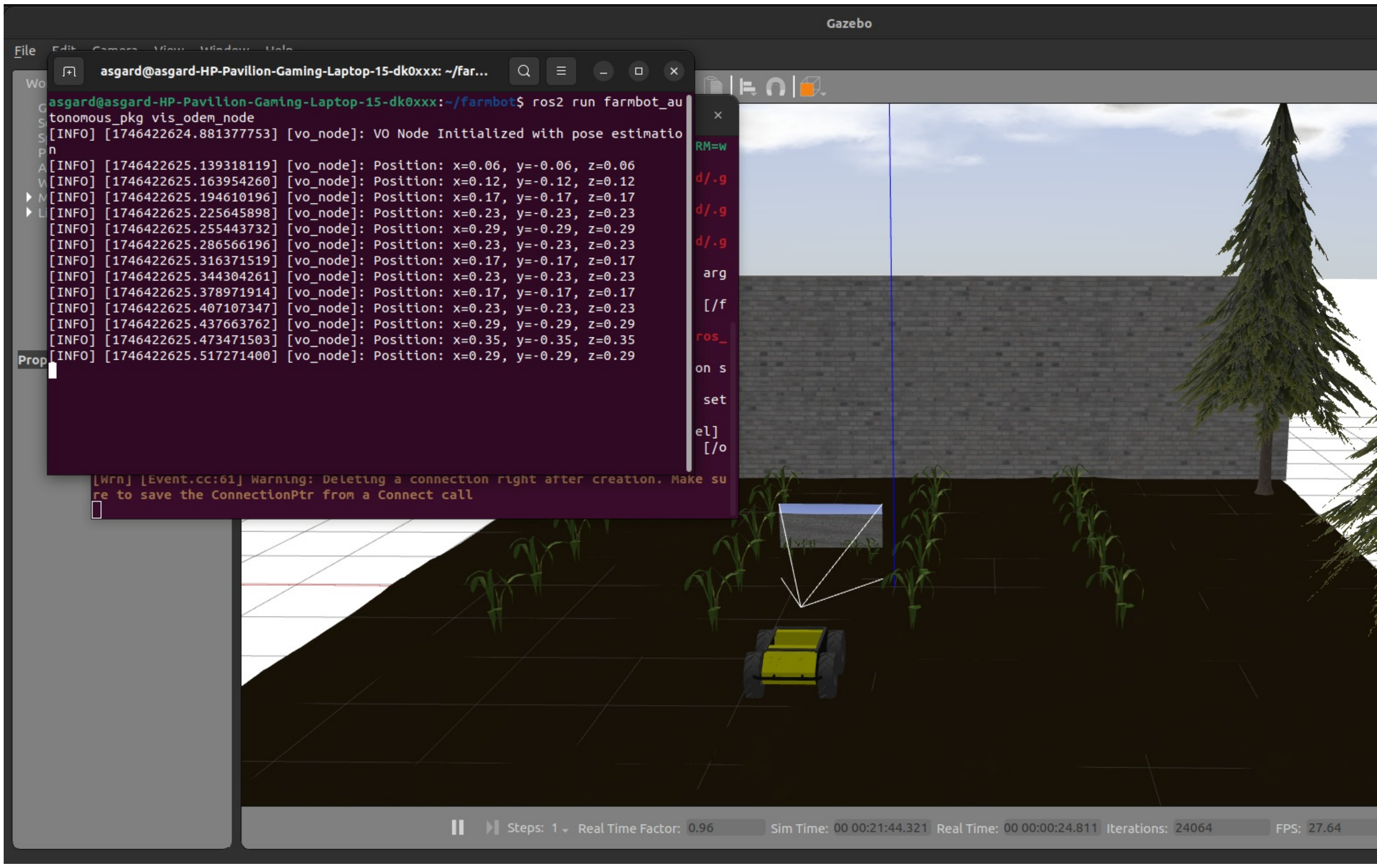
- After integrating the GNSS sensor into the Husky SDF, I encountered persistent errors during simulation indicating that the libgazebo_ros_gps.so plugin could not be loaded. Although initial checks using ls suggested the library might exist, further inspection using ldd and locate confirmed that the plugin was not actually present on the system.
- I attempted to fix the issue by exporting GAZEBO_PLUGIN_PATH and LD_LIBRARY_PATH, and verified that my environment variables were correct, but the error persisted. This made it clear that the problem wasn't path-related—it was an installation issue.
- To resolve it, I revisited the package installation and confirmed that ros-humble-gazebo-ros-pkgs and ros-humble-gps-tools had already been installed. However, even after reinstallation, the GPS plugin was still missing, suggesting it may not be bundled in the prebuilt binaries on my system.
- To address this, I cloned the gazebo_ros_pkgs repository from GitHub and built it from source using a fresh workspace targeting the ROS2 Humble branch. This ensured that all Gazebo-ROS integration plugins, including the GPS plugin, were compiled and made available.
- After the build, I sourced the new workspace and verified that the plugin was successfully found and loaded during simulation.
- I also added the workspace's setup.bash to my .bashrc for future sessions. This debugging process not only fixed the missing plugin issue but also made the GNSS module fully operational, allowing the robot to publish GPS data in the simulated environment.

Action Item 3: Debugging and Refining Visual Odometry for Monocular Camera Input – 3 hour(s).

Project Work Summary

- I worked to identify and resolve significant pose estimation errors in my monocular visual odometry (VO) pipeline implemented in ROS2 using OpenCV and Python. The rover's estimated pose was drifting erratically and did not match expected motion (e.g., moving linearly along the x-axis). To debug this:
- I reviewed the VO implementation line by line, identifying critical flaws such as the use of raw, unscaled translation vectors and the absence of inlier filtering when recovering pose from the Essential matrix.
- I corrected the pose integration step by properly normalizing and scaling the translation vector before applying it to the cumulative transformation matrix and then I added filtering logic to use only RANSAC-inlier matches for pose estimation to reduce the impact of outliers.
- I removed unnecessary components (e.g., video writers) from the pipeline to streamline testing and focus solely on pose estimation output and verified the validity of the camera intrinsic matrix by extracting fx, fy, cx, and cy parameters from the /camera_info ROS2 topic and ensuring alignment with the hardcoded intrinsics.
- I introduced structured debugging logs and trajectory storage to monitor frame-to-frame translation, inlier match counts, and trajectory drift and enforced a planar motion assumption by

zeroing vertical translation (y) to reflect the ground robot's movement constraints and minimize unobservable motion.
- I validated the corrected output visually and numerically, confirming improved stability in the estimated trajectory and a more reasonable progression of position values.



```
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/farmbot$ ros2 run farmbot_au
tonomous_pkg vis_odem_node
[INFO] [1746422624.881377753] [vo_node]: VO Node Initialized with pose estimatio
n
[INFO] [1746422625.139318119] [vo_node]: Position: x=0.06, y=-0.06, z=0.06
[INFO] [1746422625.163954260] [vo_node]: Position: x=0.12, y=-0.12, z=0.12
[INFO] [1746422625.194610196] [vo_node]: Position: x=0.17, y=-0.17, z=0.17
[INFO] [1746422625.225645898] [vo_node]: Position: x=0.23, y=-0.23, z=0.23
[INFO] [1746422625.255443732] [vo_node]: Position: x=0.29, y=-0.29, z=0.29
[INFO] [1746422625.286566196] [vo_node]: Position: x=0.23, y=-0.23, z=0.23
[INFO] [1746422625.316371519] [vo_node]: Position: x=0.17, y=-0.17, z=0.17
[INFO] [1746422625.344304261] [vo_node]: Position: x=0.23, y=-0.23, z=0.23
[INFO] [1746422625.378971914] [vo_node]: Position: x=0.17, y=-0.17, z=0.17
[INFO] [1746422625.407107347] [vo_node]: Position: x=0.23, y=-0.23, z=0.23
[INFO] [1746422625.437663762] [vo_node]: Position: x=0.29, y=-0.29, z=0.29
[INFO] [1746422625.473471503] [vo_node]: Position: x=0.35, y=-0.35, z=0.35
[INFO] [1746422625.517271400] [vo_node]: Position: x=0.29, y=-0.29, z=0.29
```

```
[Wrn] [Event.cc:61] Warning: Deleting a connection right after creation. Make su
re to save the ConnectionPtr from a Connect call
```

Gazebo
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx: ~/far...
Steps: 1
Real Time Factor: 0.96
Sim Time: 00 00:21:44.321
Real Time: 00 00:00:24.811
Iterations: 24064
FPS: 27.64

-
-

Action Item 4: GNSS-based Localization for Autonomous Vehicles: Prospects and Challenges – 3 hour(s).

Research

- https://www.researchgate.net/publication/337656003_GNSS-based_Localization_for_Autonomous_Vehicles_Prospects_and_Challenges
- GNSS-based Localization for Autonomous Vehicles: Prospects and Challenges
- Summary of Report
  - This paper reviews various classes of GNSS positioning used in autonomous vehicles, including SPS, RTK, and PPP, focusing on their accuracy, latency, and cost trade-offs. It emphasizes how these systems behave in outdoor environments, particularly when facing multipath errors, canopy occlusion, or signal degradation.
  - Sensor fusion takes center stage, especially with EKF and UKF approaches used to combine GNSS with IMU or VO data. The review highlights open-source frameworks such as ROS (and robot_localization) as key tools for implementing these fusion pipelines.
  - It discusses real-world GNSS deployment in off-road, agricultural, and urban scenarios. In agricultural environments, fused GNSS-IMU systems with RTK corrections were able to maintain ~10 cm accuracy, although signal dropouts and vegetation interference remained a challenge.
  - Coordinate transformation techniques, including UTM conversion and base_link alignment, are outlined for translating GNSS data into robot-centric frames. The paper concludes by acknowledging that standalone GNSS isn't enough for tight-path control and must be fused with local sensors to meet application-level precision.
- Relation to Project
  - The breakdown of GNSS error types directly influenced how we configured the GPS plugin in Gazebo to simulate realistic noise profiles.
  - I adopted the same ROS-native pipeline (robot_localization with EKF and navsat_transform_node) mentioned in the paper for fusing GNSS data with other motion inputs.
  - The discussion on coordinate transforms helped guide our GPS-to-UTM conversion and how to properly anchor global data to the robot's local frame during row tracking.
  - The case studies on agricultural field robots supported our project direction, showing that GNSS fusion is both necessary and validated in environments like ours.
  - Reading this made it clear that pre-validating GNSS behavior in simulation is a smart and common step before deploying anything physical in farmland.
- Motivation for Research
  - I needed to understand what GNSS could realistically offer before building an overcomplicated sensor suite — this paper gave a grounded view of strengths and limits.
  - The fusion strategies (GNSS + IMU + VO) laid out here gave me a reference architecture that matched ROS2 capabilities and my own needs.
  - I also wanted reassurance that simulating GNSS in Gazebo was worth the effort — seeing other teams validate modules that way made me more confident.
  - Finally, the review encouraged me to start designing fallback behaviors for signal dropouts early on, using VO as a dead-reckoning backup.

Action Item 5: ORB-SLAM: a Versatile and Accurate Monocular SLAM System – 3 hour(s).

Research

- https://arxiv.org/abs/1502.00956
- ORB-SLAM: A Versatile and Accurate Monocular SLAM System
- Summary of Report
  - This paper presents ORB-SLAM, a robust monocular SLAM system that performs simultaneous tracking, mapping, and loop closure in real time using only ORB features. The system is modular, consisting of parallel threads for tracking, local mapping, and loop closing, which allows it to scale efficiently.
  - ORB-SLAM handles pose drift using covisibility graphs and pose-graph optimization, enabling global consistency over large environments. It also includes a relocalization mechanism that allows recovery from tracking loss.
  - The authors benchmarked the system on diverse datasets including KITTI and TUM, showing it performs better than both direct methods and older keyframe-based pipelines.
  - The system is especially notable for its versatility — it works indoors, outdoors, and on handheld or robot-mounted setups, all with just a single RGB camera.
- Relation to Project
  - The architecture of our VO node in ROS2 draws inspiration from ORB-SLAM's front-end tracking pipeline, particularly in using ORB features and essential matrix decomposition.
  - I used their loop closure and drift-reduction ideas to understand what our VO pipeline is missing and why it needs GNSS to stay globally correct.
  - The discussion on feature sparsity and light variation was directly applicable — our farmland textures are low contrast, and I began tuning ORB thresholds and frame filters accordingly.
  - It gave me confidence that a monocular setup can get the job done if used with care, which is important given our resource and sensor constraints.
- Motivation for Research
  - I needed to understand the limitations of monocular VO before relying on it in an outdoor scenario with repetitive textures like dry soil or row crops.
  - The performance benchmarks here gave me realistic expectations and helped validate the simplified pipeline I was building in OpenCV.
  - It also pushed me to design some basic failure detection (based on inlier counts and match quality) so that the system can flag bad VO frames rather than blindly integrate them.

Action Item 6: M2C-GVIO: Motion Manifold Constraint Aided GNSS-Visual-Inertial Odometry for Ground Vehicles – 3 hour(s).

Research

- Link to Article
- Title of the Article (each research article must be its own action item)
- Summary of Report
    - This paper introduces M2C-GVIO, a filter-based sensor fusion algorithm that combines GNSS, visual-inertial odometry (VIO), and novel motion manifold constraints. The goal is to improve pose estimation accuracy and robustness for ground vehicles, especially in degraded visual or GNSS environments.
    - The authors extend the Multi-State Constraint Kalman Filter (MSCKF) framework by embedding velocity, rotation, and translation constraints that reflect how ground vehicles are confined to a surface (the “motion manifold”). The paper argues this is a better alternative than fixed kinematic assumptions like non-holonomic constraints.
    - An adaptive noise estimation technique is proposed, which tunes the manifold constraint weights based on visual/GNSS quality and vehicle motion (e.g., speed, terrain roughness). Observability analysis confirms that these constraints enhance pose reliability without sacrificing filter consistency.
    - Extensive simulation (synthetic landmark circle) and real-world tests (Brno and Kaist datasets) demonstrate that M2C-GVIO outperforms state-of-the-art algorithms like VINS-Mono and ORB-SLAM3, especially in difficult scenes like bumps, turns, and night driving. The gains are most prominent when GNSS or visual info is unreliable.
    - The fusion approach strikes a balance between computational efficiency and localization accuracy, avoiding the latency overhead of graph-based optimization techniques while remaining robust in diverse scenarios.
- Relation to Project
    - The fusion of GNSS and VIO with adaptive confidence weighting is exactly what we aim for in the Farmbot GNSS-Visual localization module. The use of motion manifold constraints (e.g., "vehicle must stay on flat terrain") fits well for row-following in agriculture.
    - Their use of a loosely coupled GNSS-VIO system inside an MSCKF structure aligns with our goal of building a ROS2-compatible, real-time-capable EKF that fuses VO and GNSS in Gazebo simulation.
    - The adaptive filtering technique, which downweights visual constraints during night scenes or poor feature matches, offers practical inspiration for our fallback strategy (e.g., prioritize GNSS when field lighting is poor).
    - The observability analysis and use of constraints to overcome unobservable DOFs is directly useful in ensuring our EKF converges when VO lacks scale or GNSS has sparse updates (e.g., during occlusion by vegetation).
    - Finally, the paper gives us insight into filter initialization techniques for GNSS-VIO, including coordinate frame alignment (ENU vs global), which is a current pain point in our ROS2 implementation.
- Motivation for Research
    - I wanted to understand how to keep GNSS-VIO fusion stable when either GNSS or camera data degrades—particularly relevant for long row farming where GNSS dropouts (e.g., tree shadows) are common and monocular VO lacks scale.
    - This paper goes beyond simple sensor fusion by modeling geometric constraints from the environment, a concept we hadn’t incorporated yet. It made me rethink how to embed assumptions like “the robot can’t fly” into our state estimator.
    - I also learned about the trade-off between loosely coupled (simpler) and tightly coupled (more accurate but complex) fusion, and how MSCKF hits a sweet spot for embedded real-time systems like ours.
    - Reading this paper gave me a better sense of how to validate a GNSS-VIO system, including simulation setup (synthetic loops with landmarks), and practical dataset benchmarks (Brno/Kaist)—things I now want to integrate into our Gazebo tests.

Action Item 7: GNSS-VO Integration and Validation – 1 hour(s).

Project Work Summary

- Run a fresh simulation to verify that GNSS messages are being published with the correct timestamp and coordinate frame and log outputs to confirm that horizontal and vertical noise are behaving as configured, and begin visualizing GPS position in RViz.
- Set up the necessary launch files and static transforms to convert /fix into /odometry/gps using UTM projection, aligning the GNSS data with base_link through robot_localization. This is the backbone of the fusion pipeline and will help clean up drift inconsistencies.
- With both VO and GNSS working independently, write and tune an EKF node that consumes both and outputs odom_combined and then simulate slow row-following and observe the filtered path using rviz2 to check if the fusion smoothens out GNSS jumps and VO drift.
- To make progress visible and comparable, collect trajectory logs from /fix, /vo/odom, and /odom_combined. Then use rqt_bag or a Python script to plot these paths and highlight improvements in drift, heading, and path smoothness after fusion.
- Prepare two repeatable test maps: one with open fields (good GNSS) and one with sparse canopy (occluded GNSS). These will help validate how the fused estimator behaves in realistic farm condition.

Action Item 8: Report Writing – 1 hour(s).

Project Work Summary

- Created word document layout to write contents of the weekly progress.
- Created relevant subsections in the epicspro website and documented 20 hours of weekly progress.
- Collected relevant documents research papers, relevant links and company's objective from their portal.