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	Debugging and Enhancing the Simulation with a Custom Rover Model	3
2	Implementing Friction Between Particle Layers	3
3	Creation and Integration of a New Rover Model	3
4	Vision-Based Navigation of Agricultural Robots: A Review	3
5	GNSS-Based Navigation for Autonomous Robots in ROS: A Practical Approach	3
6	GNSS and Vision-Based Navigation for Autonomous Robots: A Combined Approach	3
7	Next week plan	1
8	Report writing	1
	Total hours for the week:	20

Verification Documentation:

Action Item 1: Debugging and Enhancing the Simulation with a Custom Rover Model – 3 hour(s).

Project Work Summary

- Terrain Bump Placement
  - Fixed the random bump placement logic in the terrain generation step by correcting the range of `np.random.uniform` for `x_center` and `y_center`. This ensured that bumps were distributed across the entire grid instead of being concentrated at `(0, 0)`.
- Rover Initialization:
  - Adjusted the rover's initial position (`rover_center`) to dynamically place it above the terrain height (`Z.max()

+ 0.2`). This ensured that the rover was visible and not embedded inside or below the terrain.

- Persistent Rover Visualization:
  - Modified the `draw_custom_rover` function to return persistent Mayavi objects representing the rover's body and wheels. These objects were updated dynamically in each frame instead of being redrawn, improving performance and visual consistency.
- Particle Dynamics:
  - Debugged particle initialization to ensure that particles were stacked only when terrain height exceeded a threshold (`terrain_height > particle_diameter`). This prevented unnecessary particle placement in flat regions.
  - Reduced friction coefficient in `apply_layer_friction` to allow smoother particle movement while maintaining realistic interactions.
- Animation Loop Optimization:
  - Ensured that the animation loop properly updated both particle positions and rover positions using `mlab_source.set`. Removed redundant clearing of the scene (`mlab.clf()`), which caused performance issues.
  - Verified that `yield` was correctly used in the animation loop to allow Mayavi to render updates dynamically.
- Camera Settings and Visualization:
  - Adjusted camera settings using `mlab.view` to ensure all elements (terrain, particles, and rover) were visible throughout the simulation. Reduced particle visualization load by sampling every 10th particle for rendering without affecting simulation accuracy.

Action Item 2: Implementing Friction Between Particle Layers – 3 hour(s).

Project Work Summary

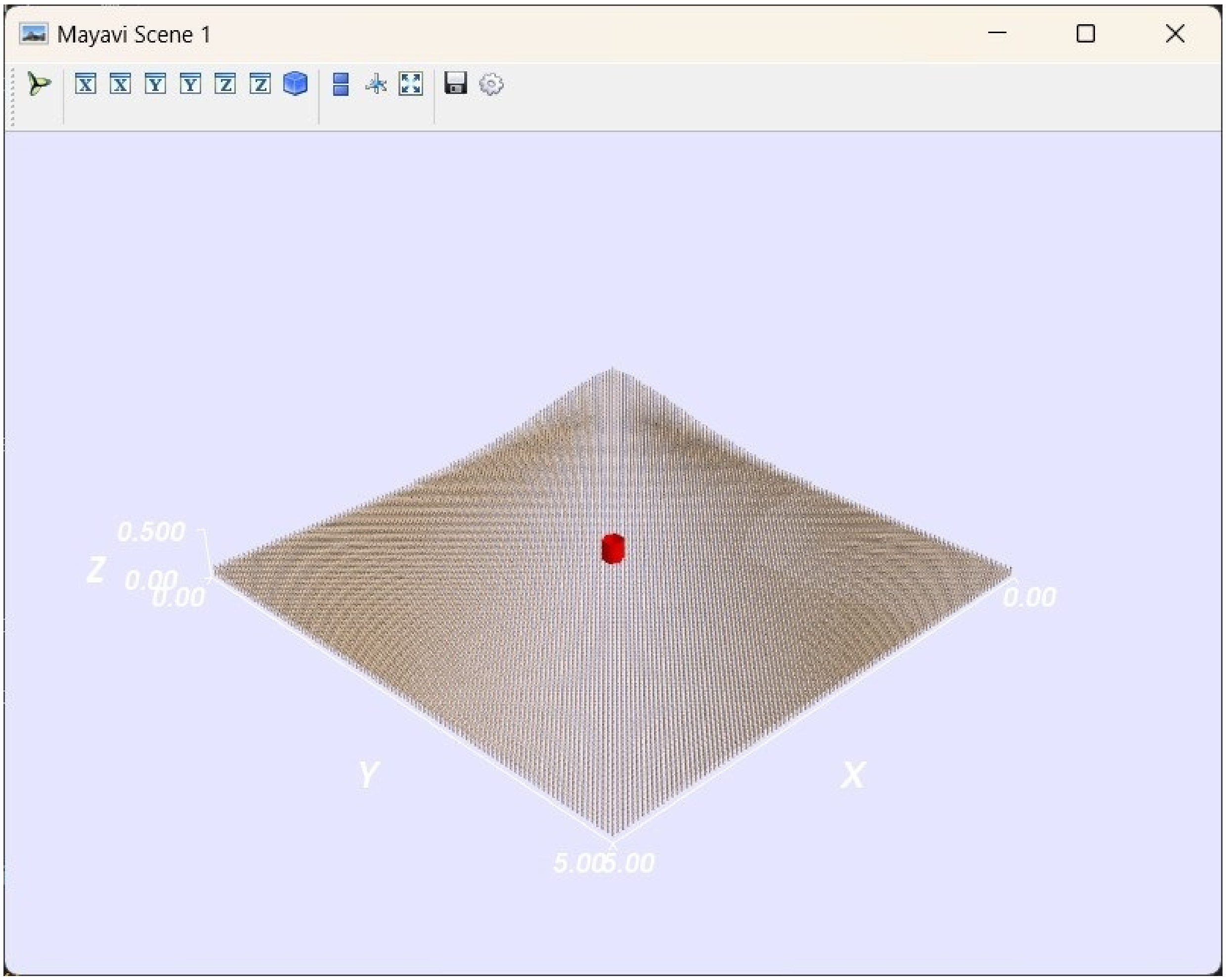
- Layer Identification:
  - Identified vertically aligned particles by comparing their x and y coordinates within a tolerance defined by their radii.
  - Determined which particle is above and which is below based on their z-coordinates.
- Relative Velocity Calculation:
  - Calculated the relative velocity between vertically aligned particles to determine the direction and magnitude of motion.
- Friction Force Application:
  - Applied a frictional force proportional to the relative velocity of the particles, scaled by a user-defined friction coefficient.
  - Reduced the velocity of the upper particle to simulate resistance from the lower particle.
- Algorithm Design:
  - Used efficient loops to iterate through all particle pairs and applied friction only to vertically aligned particles.
  - Ensured that friction was applied symmetrically and consistently across all layers.
- Numba Optimization:
  - Decorated the function with `@jit(nopython=True)` from Numba to accelerate computations for large numbers of particles.
  - Converted all inputs (positions, velocities, radii) to NumPy arrays for compatibility with Numba and efficient memory access.
- Integration with Animation:
  - Integrated the layer-based friction model into the animation loop to dynamically update particle velocities during each simulation step.
  - Verified that the friction model worked seamlessly with other components, such as collision detection and boundary checks.

Action Item 3: Creation and Integration of a New Rover Model – 3 hour(s).

Project Work Summary

- Custom Rover Design:
  - Created a new `draw_custom_rover` function to visualize the rover using Mayavi primitives.
  - Designed the rover's body as a vertical cylinder (`mlab.plot3d`) for realism.
  - Added flexibility to adjust the rover's dimensions dynamically based on its position.
- Dynamic Visualization:
  - Ensured that the `draw_custom_rover` function returned persistent Mayavi objects representing the rover's body.
  - Enabled dynamic updates to the rover's position during simulation by modifying its `mlab_source` properties.
- Replacement of Older Model:
  - Replaced the previous cuboid-based implementation with the new cylindrical-body design.

- Updated all references to the old model to use the new `draw_custom_rover` function for consistency.
- Integration with Simulation:
  - Integrated the new rover model into the animation loop, allowing it to move dynamically across the terrain.
  - Updated rover position in each frame based on velocity and ensured proper collision detection with particles.
- Algorithmic Improvements:
  - Optimized the visualization by avoiding redundant clearing and redrawing of objects in each frame.
  - Used NumPy arrays for efficient computation of positions, velocities, and interactions between particles and the rover.
- Outcome:
  - The resulting simulation features a visually appealing cylindrical-body rover that moves smoothly across the terrain.
  - The new model integrates seamlessly with existing components, maintaining real-time performance while enhancing visual realism.



Action Item 4: Vision-Based Navigation of Agricultural Robots: A Review – 3 hour(s).

Project Work Summary

- "Vision-Based Navigation of Agricultural Robots: A Review"
- J. Bac, T. Wójcik, A. Molin, P. Przybyłek, and P. Kołodziejczyk
- https://doi.org/10.3390/s21093102
- Summary of the Paper
  - This paper provides a comprehensive review of vision-based navigation techniques for agricultural robots.
  - It discusses how robots can use visual data (e.g., camera feeds, depth sensors) to navigate through complex farm environments autonomously.
  - Key Highlights
    - Vision-Based Navigation Techniques:
      - Explores methods like feature extraction, optical flow, and visual odometry.
      - Discusses the integration of stereo cameras and LiDAR for depth perception.
    - ROS Integration:

        - Highlights how ROS is used as a middleware for implementing and testing navigation algorithms.
        - Describes ROS packages like `move_base` for path planning and `image_pipeline` for processing visual data.
      - Applications in Agriculture:
        - Row-following in crops using visual markers.
        - Obstacle detection and avoidance in unstructured farm environments.
        - Mapping fields using SLAM (Simultaneous Localization and Mapping)
- Relation to Project
  - Use Visual Data for Navigation:
    - Implement a camera-based system to detect terrain features or obstacles.
    - Use OpenCV with ROS to process images and extract navigational information.
  - Integrate SLAM for Mapping:
    - Use ROS packages like `rtabmap_ros` or `gmapping` to build a map of the farm environment.
    - Combine visual data with odometry to improve localization accuracy.
  - Path Planning with Visual Cues:
    - Use `move_base` to plan paths across the field while avoiding obstacles detected from camera feeds.
    - Implement row-following algorithms using line detection or feature matching.
  - Simulation in Gazebo:
    - Create a simulated farm environment in Gazebo with rows of crops and obstacles.
    - Test the navigation algorithms using a virtual robot equipped with cameras and LiDAR.
- Motivation for Research
  - The paper provides insights into how vision-based techniques can be integrated into ROS for agricultural applications.
  - You can use these techniques to navigate your custom rover model across the simulated terrain while avoiding particles or obstacles.

Action Item 5: GNSS-Based Navigation for Autonomous Robots in ROS: A Practical Approach – 3 hour(s).

Project Work Summary

- GNSS-Based Navigation for Autonomous Robots in ROS: A Practical Approach
- D. Kümmerle, C. Stachniss, and W. Burgard
- https://doi.org/10.1109/ICRA48506.2021.9561927
- Summary of the Paper
  - This paper discusses the integration of GNSS data into the ROS framework to enable autonomous navigation of robots in outdoor environments.
  - It highlights the challenges of GNSS localization in agricultural or unstructured environments and provides solutions for combining GNSS with other sensing modalities (e.g., IMU, LiDAR) for robust navigation.
  - Key findings
    - GNSS Simulation in ROS:
      - Describes how to simulate GNSS data using tools like `gazebo_ros_gps` plugin in Gazebo.
      - Discusses the generation of realistic GNSS noise models to mimic real-world conditions.
    - Fusion with Other Sensors:
      - Explains how to fuse GNSS data with IMU and odometry using ROS packages like `robot_localization`.
      - Demonstrates how Extended Kalman Filters (EKF) can be implemented to improve localization accuracy.
    - Path Planning and Navigation:
      - Combines GNSS-based localization with ROS navigation stack (`move_base`) for path planning.
      - Uses global GNSS coordinates for long-range navigation and local sensors (e.g., cameras or LiDAR) for obstacle avoidance.
    - Applications in Agriculture:
      - Highlights the use of GNSS-based navigation for autonomous tractors and UAVs in agricultural fields.
      - Discusses row-following algorithms using GNSS waypoints combined with visual cues.
- Relation to Project
  - Simulate GNSS Data in Gazebo:
    - Use the `gazebo_ros_gps` plugin to simulate GNSS data in a virtual farm environment.
    - Add realistic noise to the simulated data for testing robustness.
  - Fuse GNSS with Other Sensors:
    - Use the `robot_localization` package to fuse GNSS, IMU, and wheel odometry data.
    - Implement an EKF or UKF (Unscented Kalman Filter) for state estimation.
  - Path Planning Using Waypoints:
    - Define global waypoints based on GNSS coordinates.
    - Use `move_base` to plan paths between waypoints while avoiding obstacles detected by LiDAR or cameras.
  - Test in Realistic Scenarios:

    - Simulate a farm environment in Gazebo with rows of crops and obstacles.
    - Test the robot's ability to navigate between predefined waypoints using GNSS data.
- Motivation for Research
  - The paper provides a clear methodology for integrating GNSS into ROS-based simulations.
  - You can use these techniques to navigate your custom rover model across a simulated farm environment using global waypoints combined with local obstacle avoidance.

Action Item 6: GNSS and Vision-Based Navigation for Autonomous Robots: A Combined Approach – 3 hour(s).

Project Work Summary

- GNSS and Vision-Based Navigation for Autonomous Robots: A Combined Approach
- M. Kamel, T. Stastny, K. Alexis, and R. Siegwart
- https://doi.org/10.1109/LRA.2017.2658940
- Summary of the Paper
  - This paper presents a novel approach for combining GNSS (Global Navigation Satellite System) and vision-based navigation for autonomous robots operating in outdoor environments.
  - It highlights the advantages of integrating GNSS data with visual odometry to improve localization accuracy, especially in agricultural or unstructured environments.
  - Key Highlights
    - GNSS Integration
      - Discusses the use of GNSS for global positioning in large-scale outdoor environments.
      - Explains how GNSS data can provide absolute positioning but suffers from noise and inaccuracies in certain conditions (e.g., multipath effects).
    - Vision-Based Navigation:
      - Explores visual odometry techniques to estimate relative motion using onboard cameras.
      - Highlights the use of feature extraction and matching algorithms (e.g., ORB, SIFT) for visual localization.
    - Sensor Fusion:
      - Combines GNSS data with visual odometry using an Extended Kalman Filter (EKF) for robust state estimation.
      - Demonstrates how sensor fusion compensates for the weaknesses of individual sensors (e.g., GNSS drift or visual odometry scaling errors).
    - Implementation in ROS:
      - Describes the implementation of the proposed method in ROS using standard packages like `robot_localization` for sensor fusion and `image_pipeline` for processing camera feeds.
      - Provides details on how to simulate GNSS and visual data in Gazebo for testing.
- Relation to Project
  - Simulate GNSS Data:
    - Use the `gazebo_ros_gps` plugin to simulate GNSS data in a virtual farm environment.
    - Add realistic noise models to mimic real-world conditions.
  - Implement Visual Odometry:
    - Use ROS packages like `rtabmap_ros` or `viso2_ros` to implement visual odometry using stereo cameras or RGB-D sensors.
    - Extract features from camera feeds and estimate relative motion between frames.
  - Fuse GNSS and Visual Data:
    - Use the `robot_localization` package to fuse GNSS, IMU, and visual odometry data.
    - Implement an EKF or UKF (Unscented Kalman Filter) for state estimation.
  - Path Planning with Combined Localization:
    - Define global waypoints using GNSS coordinates.
    - Use `move_base` to plan paths between waypoints while avoiding obstacles detected through vision-based techniques.
  - Test in Simulation:
    - Create a simulated farm environment in Gazebo with rows of crops and obstacles.
    - Test the robot's ability to navigate between predefined waypoints using fused localization data.
- Motivation for Research
  - The paper provides a clear methodology for combining GNSS and vision-based navigation techniques, which can be implemented in ROS simulations.
  - You can use these techniques to navigate your custom rover model across a simulated farm environment while leveraging both global positioning (GNSS) and local obstacle avoidance (vision).

Action Item 7: Next week plan – 1 hour(s).

Project Work Summary

- ### **1. Transition Simulation to ROS**
- **Objective**: Begin transitioning the terrain, particle, and rover simulation from Mayavi to ROS and Gazebo.
- **Tasks**:
- Install ROS Noetic (or any compatible version) on your system.
- Set up a Catkin workspace and create a new ROS package for the simulation.
- Export the terrain data (`X`, `Y`, `Z`) as a heightmap or SDF file to use in Gazebo.

---

2. Model the Rover in ROS
- **Objective**: Replace the Mayavi-based rover with a URDF (Unified Robot Description Format) model in ROS.
- **Tasks**:
- Create a URDF file for the rover with appropriate links (e.g., body, wheels).
- Add sensors like cameras or LiDAR to the rover model for navigation.
- Simulate rover movement using ROS controllers (e.g., `diff_drive_controller`).

---

3. Simulate GNSS Data
- **Objective**: Integrate GNSS simulation into the ROS environment for global navigation.
- **Tasks**:
- Use the `nmea_navsat_driver` package to simulate GNSS data in ROS.
- Add noise models to mimic real-world GNSS inaccuracies (e.g., signal drift, multipath effects).
- Combine GNSS data with IMU readings using the `robot_localization` package.

---

4. Implement Visual SLAM
- **Objective**: Use visual SLAM techniques for local navigation and mapping.
- **Tasks**:
- Integrate ORB-SLAM2 or RTAB-Map into your ROS package to process camera feeds and generate maps.
- Test visual SLAM in a simulated Gazebo environment with realistic terrain.

---

5. Path Planning Using Combined Data
- **Objective**: Plan paths using GNSS coordinates for global navigation and SLAM-generated maps for obstacle avoidance.
- **Tasks**:
- Implement A* or Dijkstra's algorithm for waypoint-based navigation.
- Dynamically replan paths based on obstacles detected by SLAM.

---

6. Simulate in Gazebo
- **Objective**: Create a realistic farm-like environment in Gazebo and test your system.
- **Tasks**:
- Design a Gazebo world with uneven terrain, crop rows, and obstacles like rocks or fences.
- Simulate rover movement across the farm using GNSS waypoints and SLAM-generated maps.

Suggested Workflow
1. Start by setting up a basic Gazebo simulation with a flat terrain and a simple rover model.
2. Gradually integrate GNSS data, visual SLAM, and path planning into the simulation.
3. Test each component individually before combining them into a complete system.

Expected Outcome
By the end of next week, you should have:
1. A basic simulation running in ROS/Gazebo with terrain and rover models.
2. GNSS data simulated in ROS and integrated with IMU readings for localization.
3. Initial steps towards implementing visual SLAM for mapping and navigation.

Let me know if you'd like help with any specific part of this workflow!

Citations:
[1] https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/28784843/4d56152b-ba6a-4ec6-bf0b-

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

Project Work Summary

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