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.

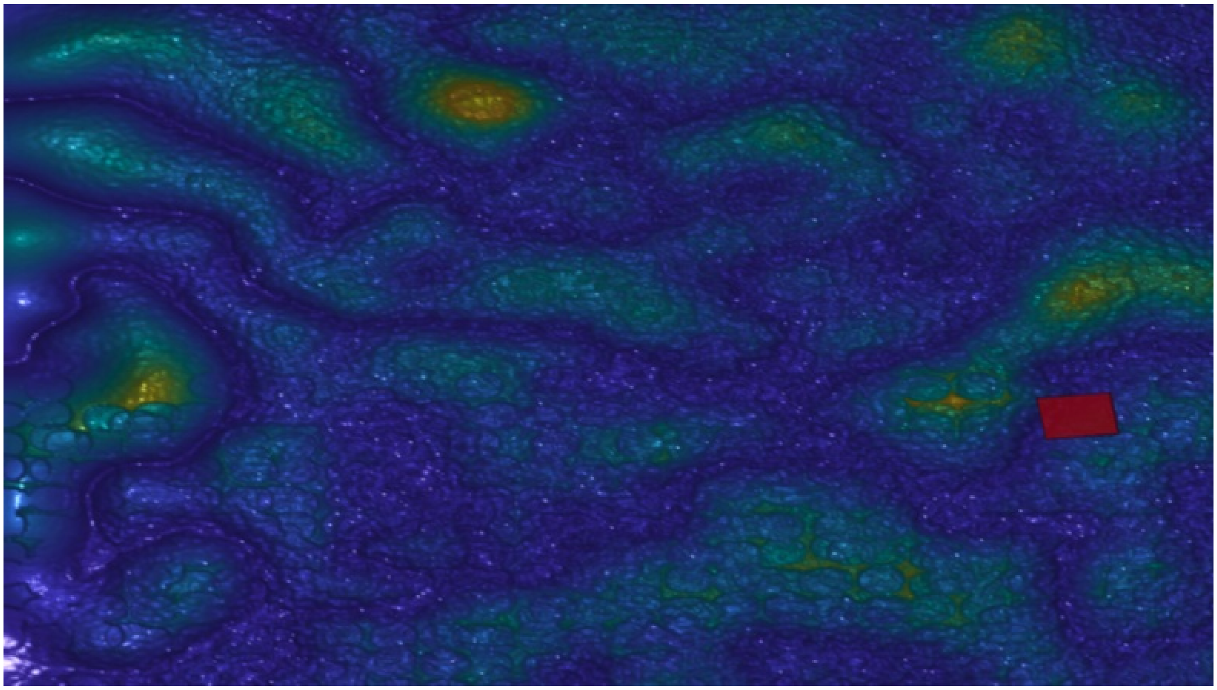
#	Action Item/ Explanation	Total Time This Week (hours)
1	Debugging grid to path conversion.	3
2	Path Smoothing Algorithm Summary	3
3	Modular Rover Class Definition for Scalable Multi-Robot Agricultural Simulations	3
4	Debugging the visual representation of the rover and expanded the terrain map	3
5	Implementing Data Collection with the new rover function	3
6	Multi-Rover Data Collection System Implementation	3
7	Next week plans	1
8	Report Writing	1
	Total hours for the week:	20

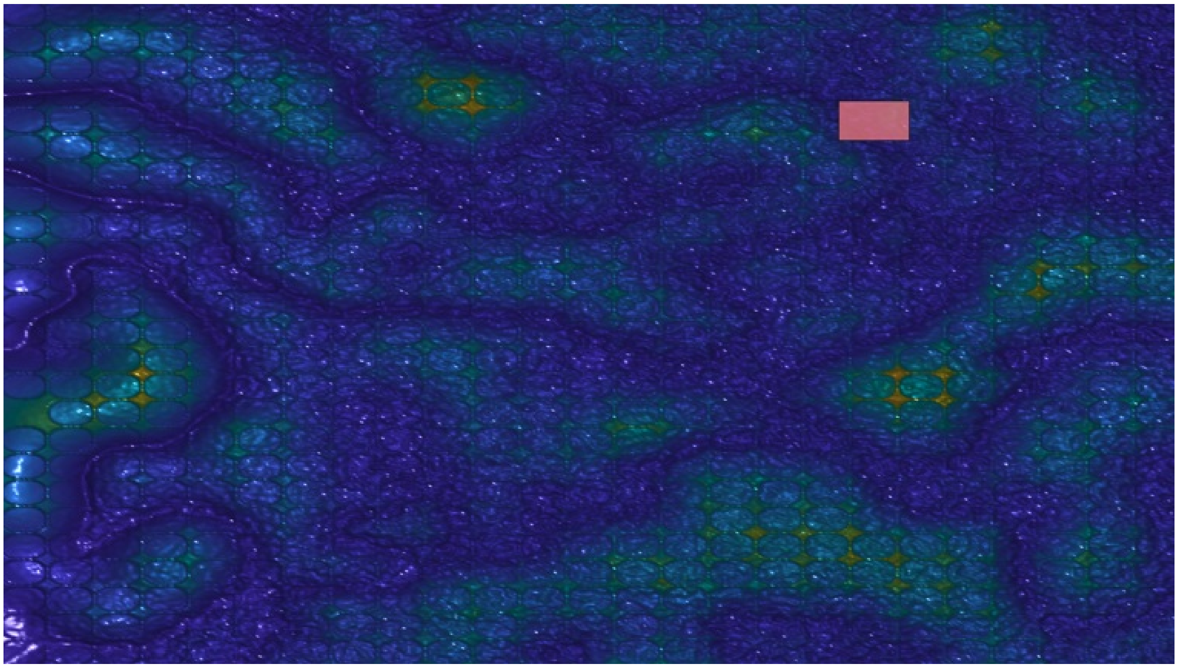
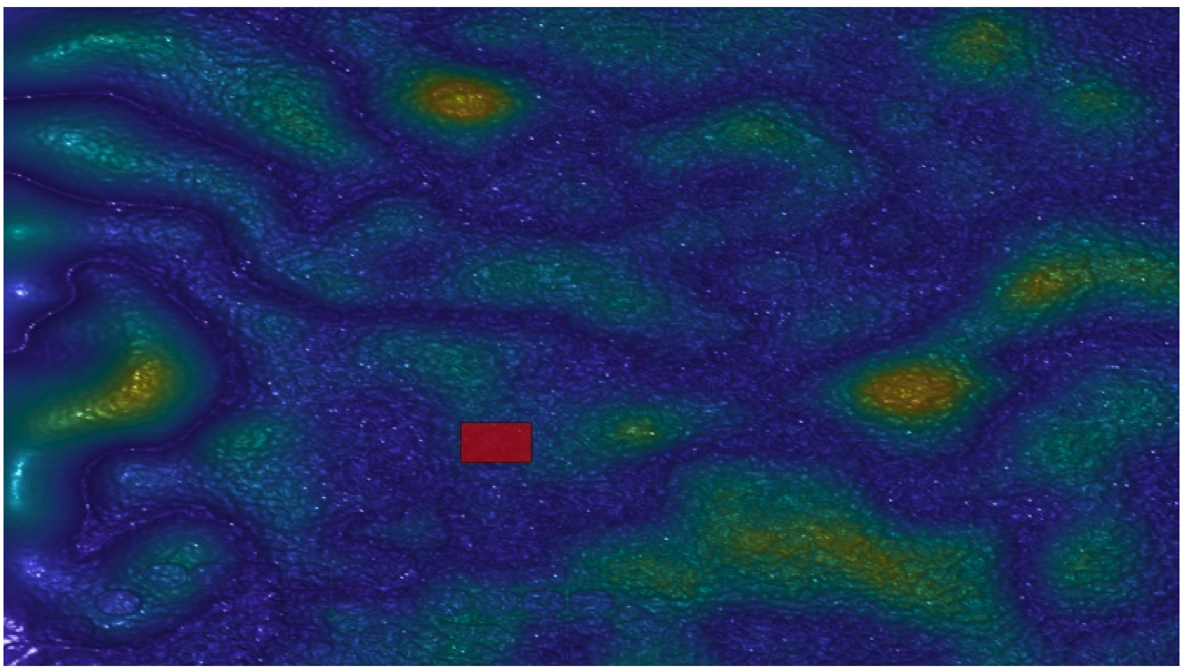
Verification Documentation:

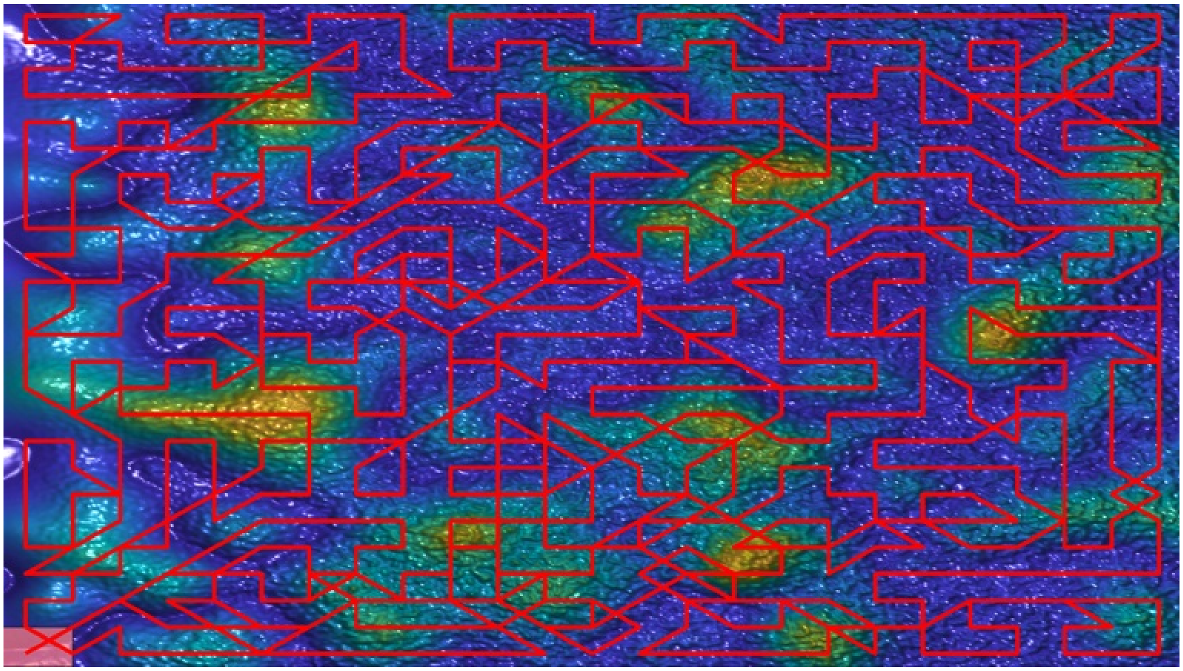
Action Item 1: Debugging grid to path conversion. – 3 hour(s).

Project Work Summary

- The previous codes collected height data in avg_height_grid which was used by A* algorithm for path planning. However, there was a mismatch between:
  - The terrain grid (high resolution: 0.01m spacing)
  - The planning grid (coarser resolution: based on plow radius)
- Grid Resolution Calculation

    - grid_resolution = floor(5 / (2 * plow_radius));
    - cell_size = 5 / grid_resolution;
    - This calculates how many cells fit in the 5x5 meter terrain
    - Each cell is sized based on the plow diameter (2 * plow_radius)
    - `cell_size` gives the actual physical size of each grid cell
- Coordinate Conversion
    - Old conversion (incorrect)
        - path_x = (path(:,2) - 1) * (2 * plow_radius);
        - path_y = (path(:,1) - 1) * (2 * plow_radius);
    - New conversion (correct)
        - path_x = (path(:,2) - 0.5) * cell_size;
        - path_y = (path(:,1) - 0.5) * cell_size;
    - Added `-0.5` to center the path within grid cells
    - Uses `cell_size` instead of direct plow radius multiplication
    - Ensures proper scaling between grid indices and physical coordinates
- Boundary Safety
    - Added boundary checking
        - path_x = max(0, min(5, path_x));
        - path_y = max(0, min(5, path_y));
    - Prevents the path from going outside the terrain bounds
    - Ensures interpolation will work correctly
- These modifications create a proper bridge between:
    - The discrete grid used by A* algorithm
    - The continuous physical space where the rover operates
    - The high-resolution terrain data used for visualization and height calculations
- The result is a more accurate and reliable path that properly corresponds to the terrain data collected in the first phase of the operation.
- 

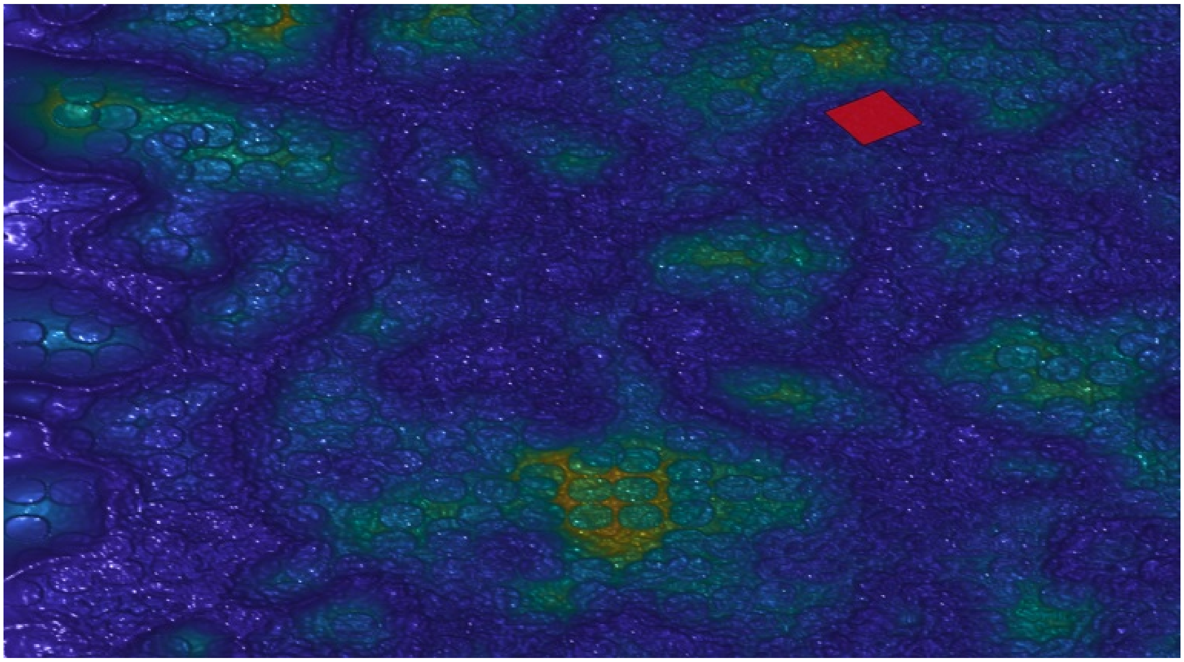
- 
- 

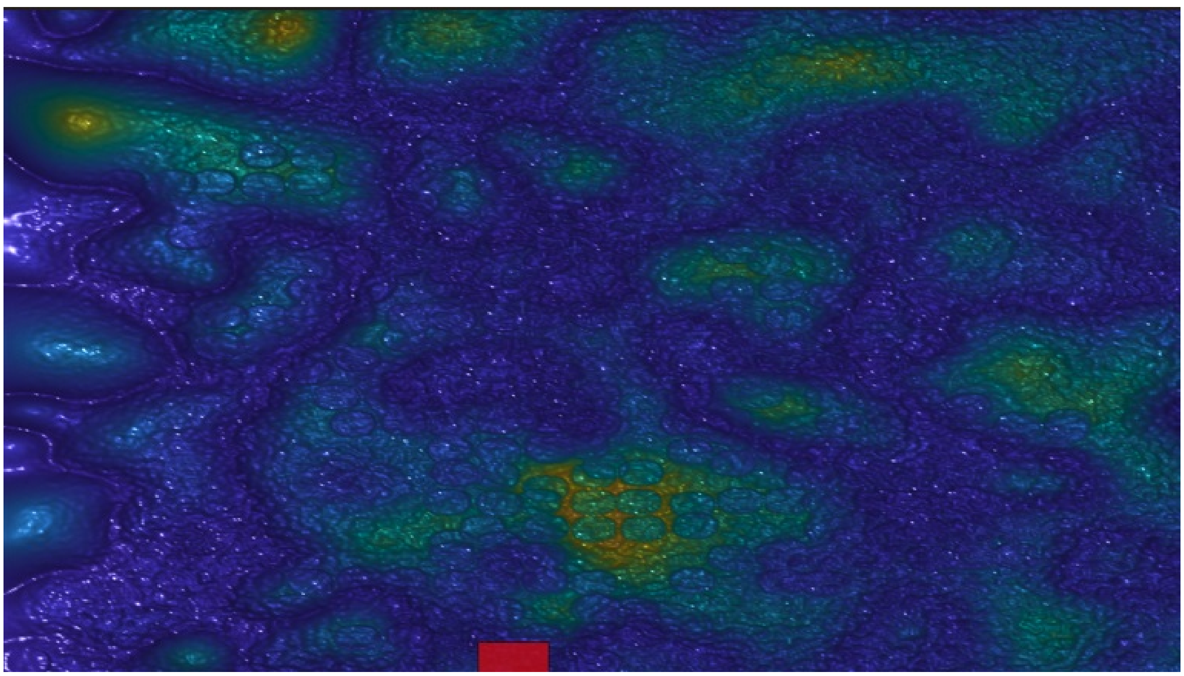
- 
-

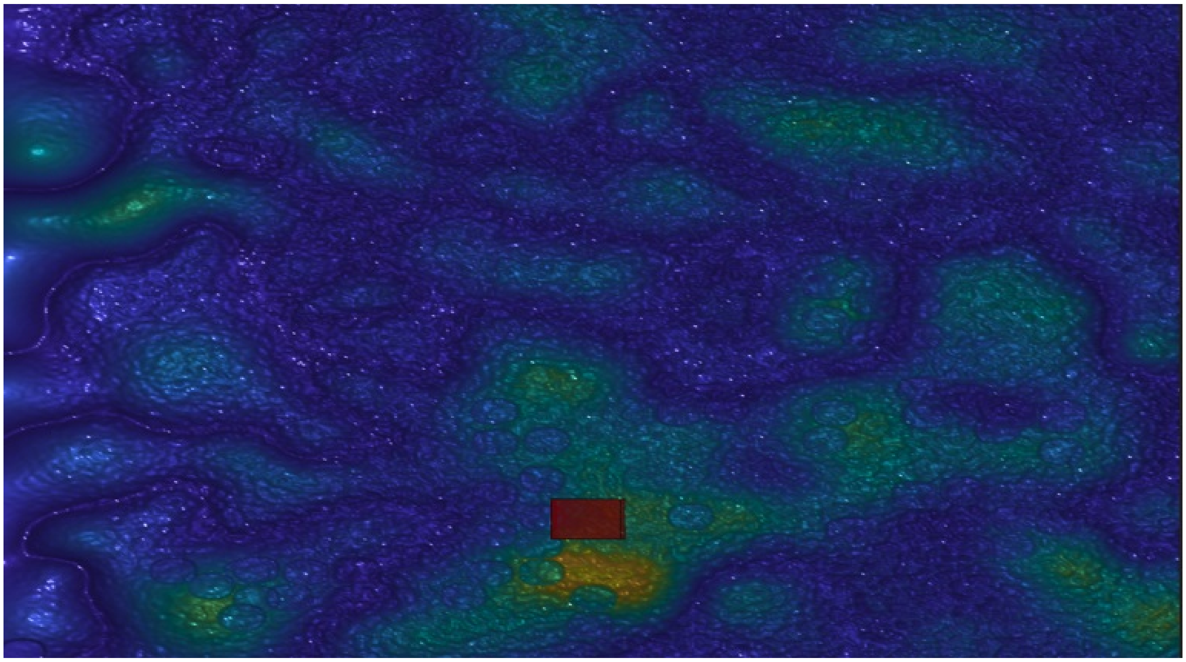
Action Item 2: Path Smoothing Algorithm Summary – 3 hour(s).

Project Work Summary

- Grid Path Processing
  - The algorithm transforms a discrete grid-based path into a smooth, continuous trajectory suitable for rover navigation.
- Implementation Components
- Path Interpolation
  - X-Coordinate Smoothing
    - smooth_path_x = interp1(1:length(path_x), path_x, linspace(1, length(path_x), num_points), 'spline');
    - Converts discrete x-coordinates into a continuous spline curve
    - Generates `num_points` evenly spaced points along the path
    - Uses cubic spline interpolation for smooth transitions
  - Y-Coordinate Smoothing
    - smooth_path_y = interp1(1:length(path_y), path_y, linspace(1, length(path_y), num_points), 'spline');
    - Applies identical smoothing process to y-coordinates
    - Maintains synchronization with x-coordinate interpolation
    - Ensures consistent spatial resolution
  - Z-Coordinate Calculation
    - smooth_path_z = interp2(X, Y, Z, smooth_path_x, smooth_path_y);
    - Maps smoothed (x,y) coordinates onto the terrain height map
    - Provides continuous height values for the rover's trajectory
    - Ensures proper vertical positioning during movement
- Operational Benefits
  - Motion Control
    - Eliminates sharp turns and sudden movements
    - Creates natural, fluid rover movement
    - Reduces mechanical stress on the rover system
  - Path Quality
    - Transforms grid-based path into continuous trajectory
    - Maintains terrain-following capability
    - Optimizes rover's navigation efficiency
  - Simulation Enhancement
    - Improves visual representation
    - Enables realistic rover behavior
    - Supports smooth animation rendering
  - This smoothing process transforms the discrete A* path into a practical, executable trajectory that considers both the rover's physical constraints and the terrain's characteristics.







Action Item 3: Modular Rover Class Definition for Scalable Multi-Robot Agricultural Simulations – 3 hour(s).

Project Work Summary

- We developed a modular Rover class in MATLAB to facilitate the transition from single-robot to multi-robot agricultural simulations. The key aspects of this implementation include:
  - Encapsulation of rover properties and behaviors:
    - The class stores essential attributes such as position, dimensions, speed, and plow radius.
    - It maintains its own visualization data, including vertices and faces for 3D rendering.
  - Path-following capabilities:
    - The Rover class can accept a predefined path and follow it autonomously.
    - It keeps track of its current position within the path and can report when it has completed the assigned route.
  - Data collection functionality:
    - The class includes methods for collecting and storing height data as the rover moves across the terrain.
  - Modular design for scalability:
    - The class is designed to be instantiated multiple times, allowing for easy expansion to multi-robot scenarios.
    - Each rover instance operates independently, facilitating parallel simulations of multiple units.
  - Separation of concerns:
    - The Rover class focuses solely on its own movement and data collection, leaving terrain interactions and modifications to be handled externally.
    - This design allows for greater flexibility in implementing different terrain modification algorithms without altering the core Rover class.
  - Efficient visualization updates:
    - The class pre-computes its geometry and updates only the necessary transformations during movement, optimizing performance for multi-robot simulations.
  - Simplified interface:
    - The class provides straightforward methods for setting paths, moving the rover, and retrieving collected data, making it easy to integrate into larger simulation frameworks.
- This modular Rover class serves as a foundation for scalable agricultural simulations, allowing researchers to easily expand from single-robot to multi-robot scenarios while maintaining consistent behavior and efficient performance across all instances.

Lunar_base_layout.m | create_3D_lunar_base.m | astar_path_following.m | Data_Collection.m | Rover.m

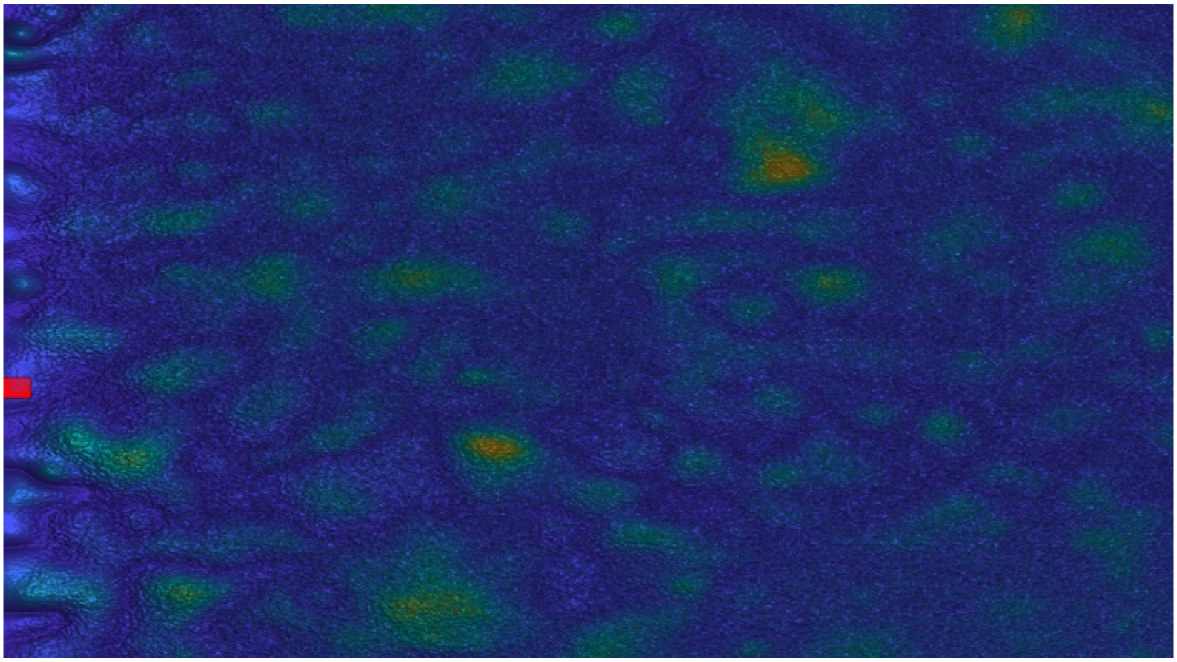
```matlab
        PlowRadius
        Patch % Handle to the rover's patch object
        HeightData % Store collected height data
        Vertices % Store the rover's vertices
        Faces % Store the rover's faces
        Path % Store the path for the rover to follow
        CurrentPathIndex % Keep track of the current position in the path
    end

    methods
        function obj = Rover(initial_position, dimensions, speed, plow_radius)
            obj.Position = initial_position;
            obj.Dimensions = dimensions;
            obj.Speed = speed;
            obj.PlowRadius = plow_radius;
            obj.HeightData = [];
            [obj.Vertices, obj.Faces] = obj.generateRoverGeometry();
            obj.Patch = patch('Vertices', obj.Vertices, 'Faces', obj.Faces, ...
                'FaceColor', 'red', 'EdgeColor', 'black', 'FaceAlpha', 0.7);
            obj.CurrentPathIndex = 1;
            obj.updateVisualization();
        end

        function setPath(obj, path)
            obj.Path = path;
            obj.CurrentPathIndex = 1;
        end
```

Action Item 4: Debugging the visual representation of the rover and expanded the terrain map – 3 hour(s).

Project Work Summary

- Rover Visualization Enhancement
  - Color Correction:
    - Addressed the issue of the rover appearing white instead of red due to lighting effects.
    - Implemented material property adjustments to ensure the rover's intended color is visible.
  - Lighting and Material Modifications:*
    - Updated the `Rover` class to include customizable material properties.
    - Modified the `updateVisualization` method to apply consistent material settings during movement.
    - Adjusted ambient, diffuse, and specular strengths to achieve the desired visual effect.
  - Constructor Update:
    - Enhanced the `Rover` constructor to initialize the patch object with appropriate material properties.
    - Ensured consistent appearance of the rover throughout the simulation.
- Terrain Map Expansion
  - Increased Map Dimensions:
    - Expanded the terrain map from 5x5 meters to 10x10 meters.
    - Modified the `meshgrid` function call to generate a larger grid while maintaining resolution
  - Terrain Feature Scaling:
    - Adjusted the number and distribution of terrain bumps to maintain feature density in the larger area.
    - Increased the number of bumps from 2000 to 8000 to populate the expanded terrain.
  - Terrain Generation Parameters:
    - Fine-tuned frequency variations for sinusoidal terrain generation to create smoother variations over the larger area.
    - Adjusted the Gaussian filter parameters to ensure appropriate smoothing for the expanded terrain.
  - Visualization Adjustments:
    - Updated axis limits to accommodate the larger 10x10 meter grid.

- Modified lighting and camera settings to ensure proper visualization of the expanded terrain.



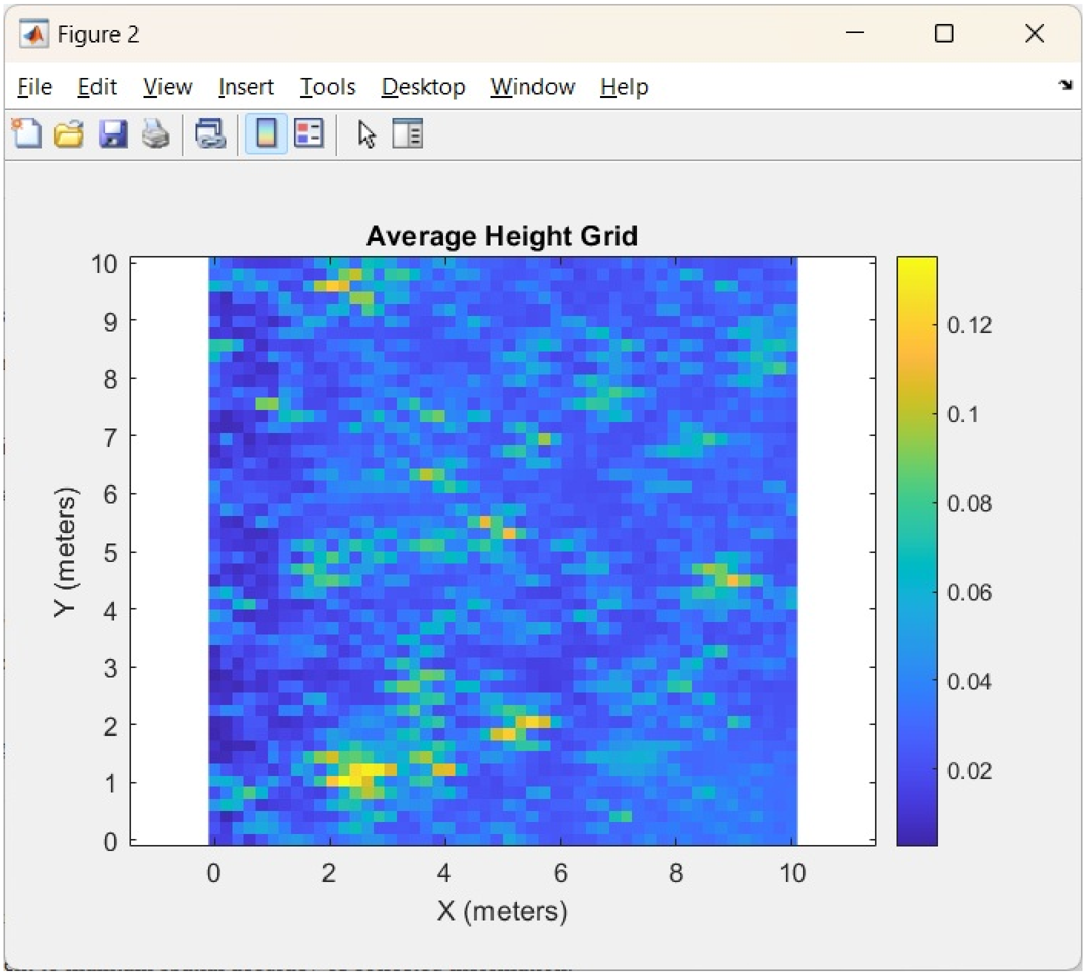
Action Item 5: Implementing Data Collection with the new rover function – 3 hour(s).

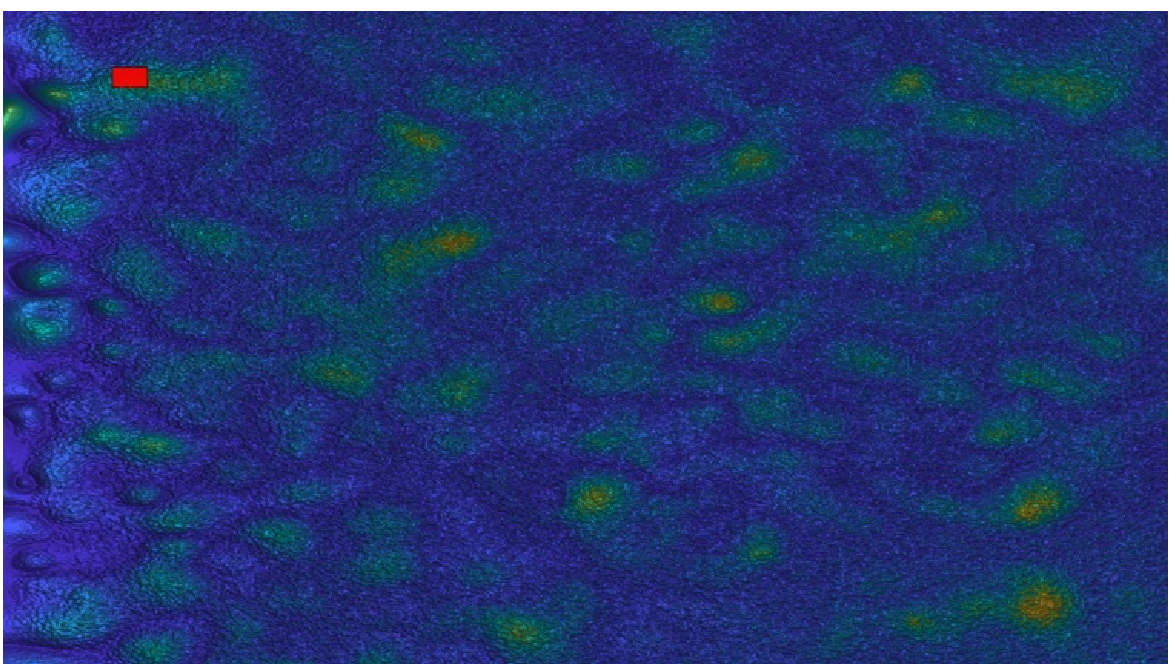
Project Work Summary

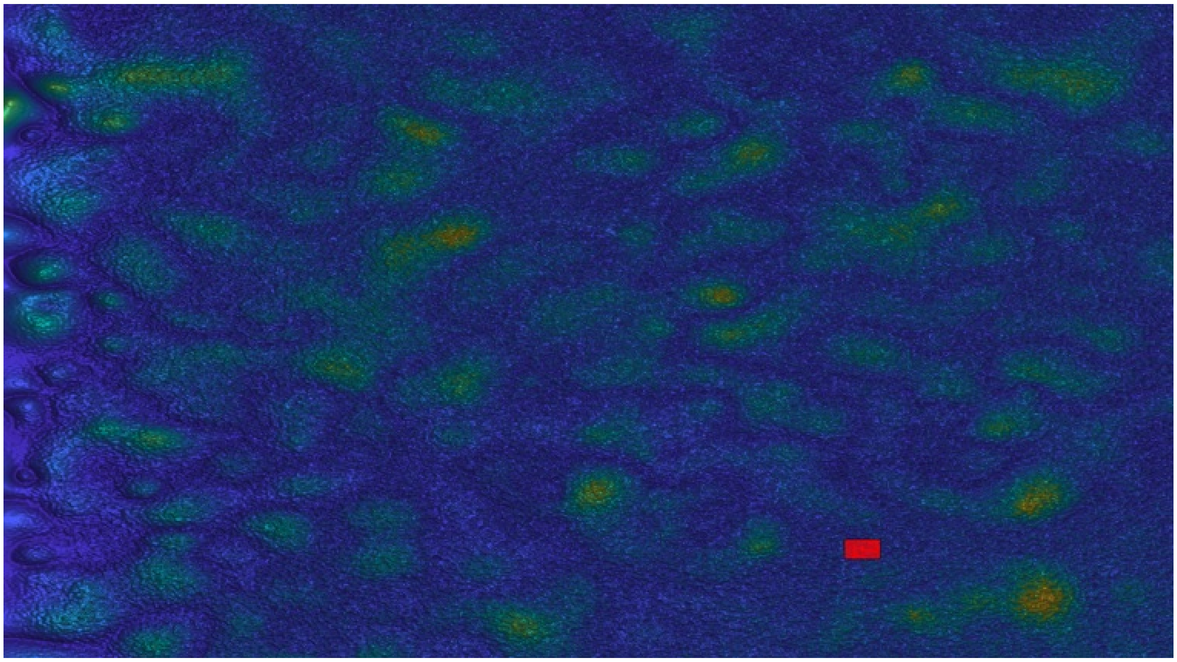
- Rover-based Data Collection
  - Height Data Acquisition:
    - Implemented a `collectHeight` method in the `Rover` class to sample terrain height at the rover's current position.
    - Utilized interpolation techniques to accurately determine terrain height from the discretized terrain model.
  - Local Terrain Sampling:
    - Extended data collection to cover a circular area around the rover, defined by its plow radius.
    - Employed a mask to isolate relevant data points within the rover's operational range.
  - Data Storage Mechanism:
    - Added a `HeightData` property to the `Rover` class to store collected terrain information.
    - Structured data storage to include position coordinates and corresponding height values.
- Gradient-based Movement
  - Terrain Gradient Calculation:
    - Implemented gradient computation using MATLAB's `gradient` function to determine local slope.
    - Integrated gradient information into the rover's movement decisions for more realistic terrain navigation
  - Adaptive Path Following:
    - Modified the `move` method to incorporate gradient information, allowing the rover to align with the terrain slope.
    - Implemented normalization of gradient vectors to ensure consistent movement speed across varying slopes.
- Data Processing and Visualization
  - Grid-based Data Aggregation:
    - Developed a system to aggregate collected height data into a grid structure.
    - Implemented averaging mechanisms to handle multiple data points within the same grid cell.
  - Visualization of Collected Data:
    - Created a separate visualization for the collected height data, distinct from the terrain rendering.
    - Utilized MATLAB's `imagesc` function to generate a color-coded representation of the collected height information.
- Integration with Simulation Loop
  - Continuous Data Collection:
    - Integrated the data collection process into the main simulation loop, ensuring continuous sampling as the rover moves.
    - Synchronized data collection with the rover's movement to maintain spatial accuracy of collected

information.

- Performance Optimization:
  - Implemented efficient data storage and processing methods to minimize computational overhead during simulation.
  - Balanced the frequency of data collection with simulation performance to ensure smooth execution.



- 

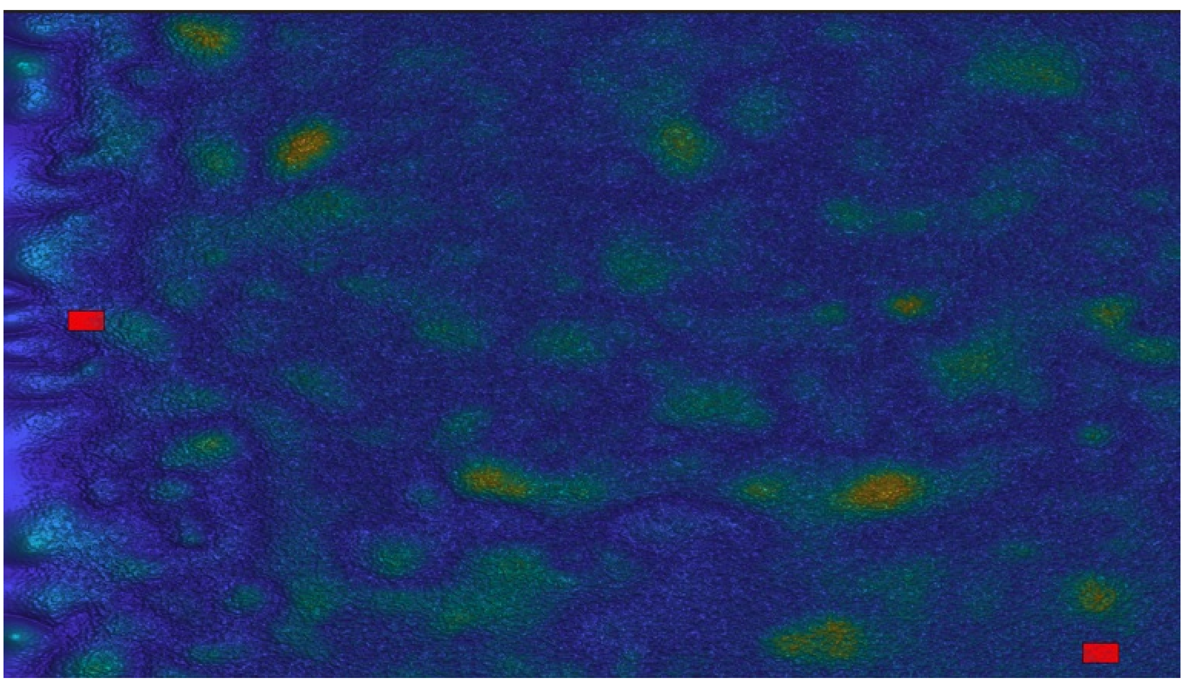


-
-

Action Item 6: Multi-Rover Data Collection System Implementation – 3 hour(s).

Project Work Summary

- We successfully enhanced our agricultural simulation environment to incorporate a multi-rover data collection system. This advancement allows for more efficient and comprehensive terrain mapping.



1. Scalable Rover Array:
- Implemented a Rover class array, allowing easy instantiation and management of multiple rovers.
- Demonstrated the system with two rovers, but designed for easy expansion to more units.

2. Path Distribution:
- Developed an algorithm to split the original path into multiple segments.
- Assigned unique path segments to each rover, ensuring complete coverage of the terrain.

3. Simultaneous Operation:

- Created a simulation loop that moves all rovers concurrently.
- Implemented checks to ensure each rover operates only within its assigned path.

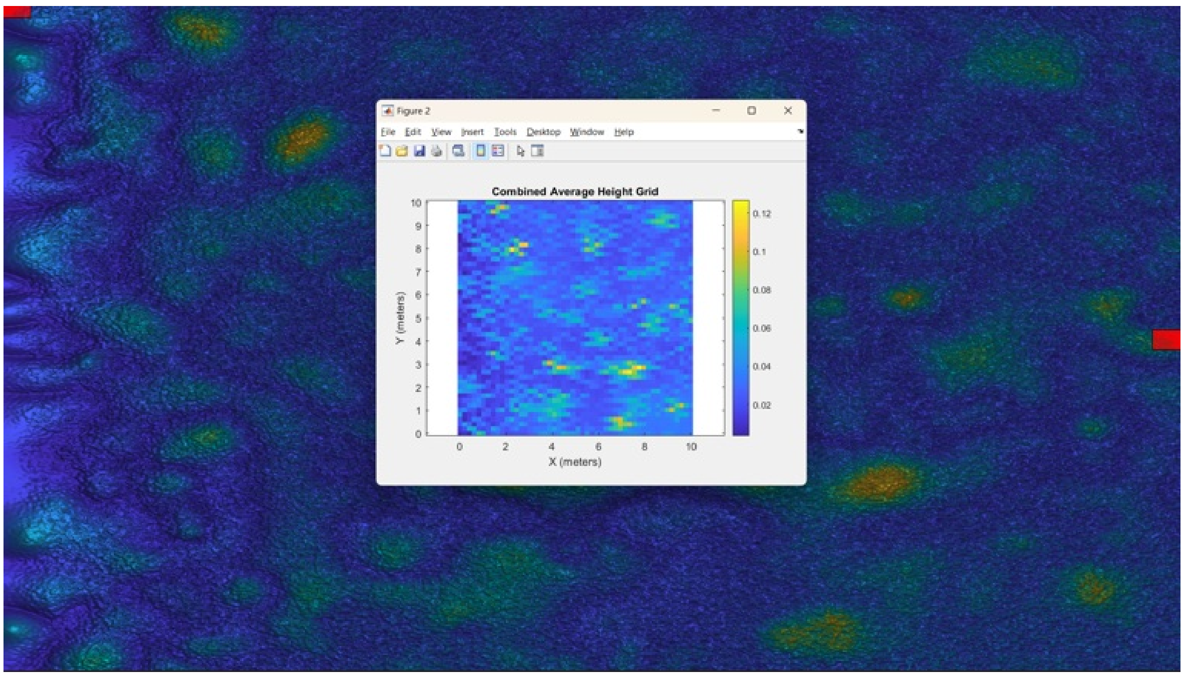
4. Unified Data Collection:
- Enhanced the Rover class to collect and store height data during movement.
- Implemented a method to combine data from all rovers into a single dataset.

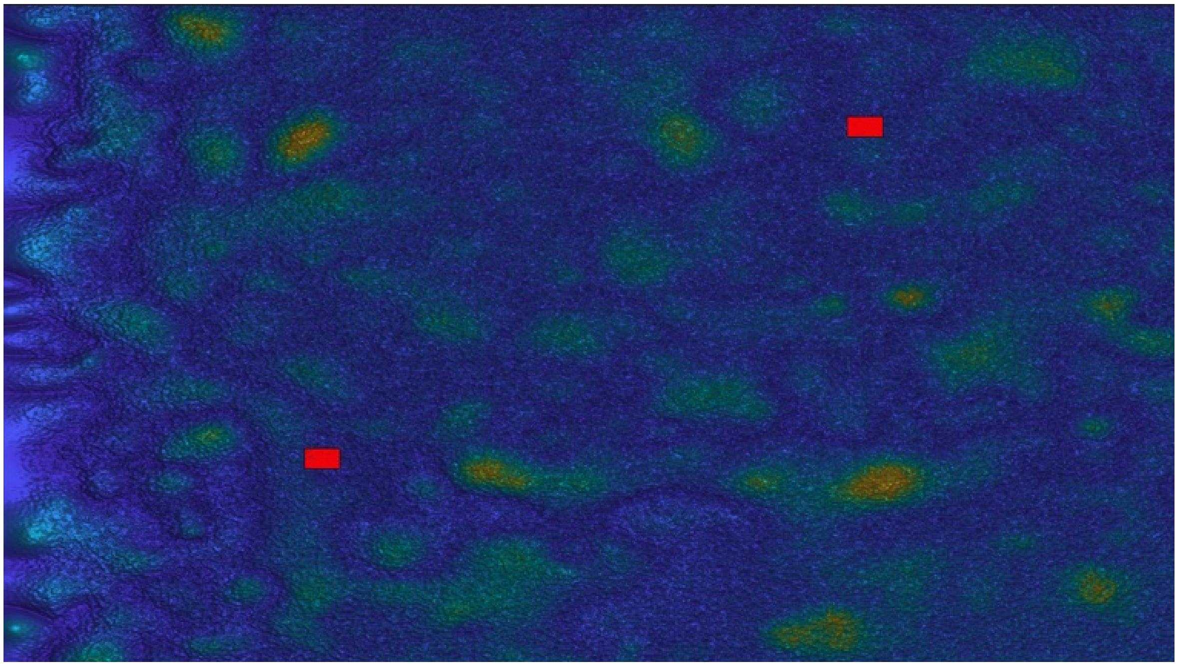
5. Efficient Simulation Control:
- Utilized array operations and loop structures to manage multiple rovers efficiently.
- Implemented a condition to terminate the simulation when all rovers complete their paths.

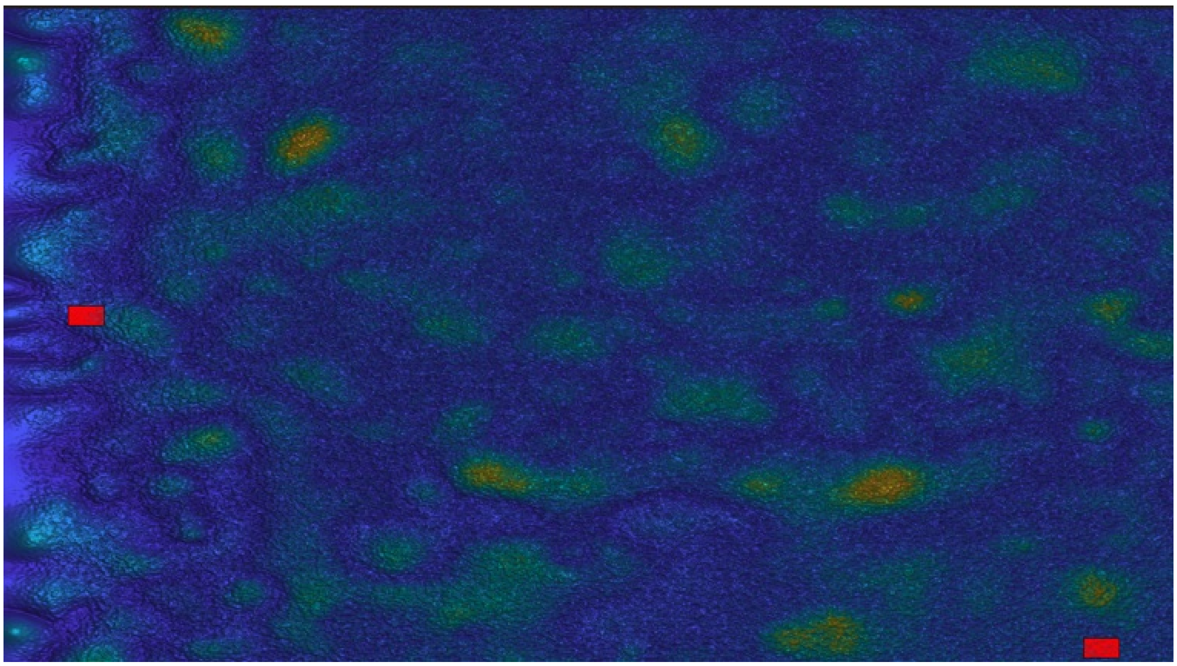
6. Visualization Enhancements:
- Updated the visualization system to display multiple rovers simultaneously.
- Maintained real-time updates of rover positions and collected data.

7. Data Processing and Analysis:
- Developed a system to aggregate and average height data from all rovers.
- Created a visualization of the combined height data, representing the complete mapped terrain.

8. Scalability and Flexibility:
- Designed the system to easily accommodate additional rovers with minimal code changes.
- Structured the code to allow for future implementation of heterogeneous rover teams with varying capabilities.







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

Project Work Summary

- We are developing an advanced multi-robot system for terrain leveling using the A* algorithm. This implementation aims to efficiently cover and level large areas of terrain using multiple autonomous robots. Key aspects of this project include:
    - Algorithm Development
      - Implement a multi-robot A* pathfinding algorithm optimized for terrain leveling tasks.
      - Develop a method to divide the terrain into sections for efficient coverage by multiple robots.
      - Incorporate terrain height data into the A* heuristic function to prioritize areas needing leveling.
    - Robot Coordination
      - Design a centralized control system to manage multiple robots simultaneously.
      - Implement collision avoidance strategies to prevent robot-robot interference.
      - Develop a dynamic task allocation system to assign robots to different terrain sections.
    - Terrain Mapping and Analysis
      - Create a system for real-time terrain mapping and height data collection.

- Implement algorithms to analyze terrain data and identify areas requiring leveling.
    - Develop a method to update the terrain model as leveling progresses.
- Path Optimization
    - Optimize robot paths to minimize travel time and maximize leveling efficiency.
    - Implement adaptive path planning to respond to changing terrain conditions.
    - Develop strategies for handling obstacles and impassable terrain sections.
- Simulation and Testing
    - Create a detailed simulation environment for testing the multi-robot system.
    - Implement various terrain scenarios to evaluate algorithm performance.
    - Develop metrics for assessing leveling efficiency and coverage completeness.
- Data Integration and Visualization
    - Design a system to integrate data from all robots into a unified terrain model.
    - Develop a real-time visualization tool to monitor leveling progress and robot positions.
    - Implement data logging and analysis tools for post-operation evaluation.

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.