Acknowledgements

I would like to express my sincere gratitude to SAC ISRO, Ahmedabad for providing me with the opportunity to work on this project, ”Image Simulation and Visualization.” This experience has been invaluable in enhancing my knowledge of satellite imaging and digital image processing.

I extend my heartfelt thanks to my industry mentor, Mr. K. Suresh, for his continuous guidance, encouragement, and technical insights that helped shape this project. His expertise in satellite imaging and simulation provided me with a strong foundation to develop this system.

I am also immensely grateful to my institute mentor, Ms. Krishna Rauji, for her unwavering support, valuable feedback, and constructive suggestions throughout the project. her mentorship played a crucial role in refining my approach and ensuring the project’s successful completion.

Lastly, I would like to thank my colleagues, peers, and everyone who supported and motivated me throughout this journey. This project would not have been possible without their constant encouragement and collaboration.

Harshalkumar SunilKumar Patel
AI and AIDS, PIET
Parul University,
Vadodara

Abstract

The Image Simulation and Visualization project aims to assist in designing and evaluating satellite imaging systems by simulating how a camera would capture images in a lunar environment. By allowing users to define various camera parameters such as position, altitude, roll, yaw, pitch, and lens settings, this system generates simulated images that closely represent the expected results from actual lunar missions.

The project is developed using PyQt5 and Qt Designer for the graphical user interface, while NumPy, PyQtGraph, and GDAL handle image processing and spatial data operations. The iterative waterfall model is followed to ensure systematic development and refinement based on feedback at each stage.

This system plays a crucial role in optimizing camera settings before deployment on lunar satellites, reducing experimental costs and improving mission accuracy. The feasibility and effectiveness of the simulation ensure that future moon exploration missions can be planned with greater precision and confidence.

Table of Contents

List of Tables

List of Figures

3 INTRODUCTION TO PROJECT

3.1 PROJECT SUMMARY

This project focuses on Moon Surface Image Simulation using satellite imaging technologies to simulate and visualize images of the moon's surface. The goal is to understand how a camera will capture an image in an unknown environment, such as the moon's surface, and help in designing and setting up camera parameters for such simulations. The project aims to provide an effective system for simulating images by adjusting parameters like camera position, altitude, roll, yaw, and pitch, which can be used in designing cameras for future lunar missions.

By simulating the moon's surface images based on user input, the system can assist in camera parameter optimization for space missions. It provides an interface where users can input pre-captured images and define specific parameters, and in return, they will get a simulated moon surface image.

3.2 PURPOSE

The primary purpose of this project is to simulate and visualize how the moon's surface will appear based on different camera parameters such as position, altitude, and orientation. The system will serve as a tool for space scientists and engineers involved in the design and calibration of lunar imaging systems.

By offering an interactive platform, the project aims to:

- Provide a realistic simulation of moon surface images based on camera position and settings.
- Aid in studying how camera will capture the images.

3.3 OBJECTIVE

The main objectives of this project are:

- To design a system that can take user input images (pre-captured) and simulate the appearance of the moon's surface under different camera parameters.

- To help in studying for lunar missions by visualizing how the camera would capture images of the moon from different perspectives.
- To develop an intuitive user interface using PyQt5 that allows users to interact with the simulation and adjust parameters like latitude, longitude, altitude, and camera orientation.

3.4 SCOPE (WHAT IT CAN AND CAN'T DO)

Scope of the Project:

- The system can simulate images of the moon's surface, enabling users to visualize how different camera parameters impact the final image.
- The tool will allow users to input camera settings, such as camera position, altitude, and orientation parameters, and generate simulated images of the lunar surface.

What the Project Can't Do:

- It cannot provide real-time image capturing or real-world data from lunar surface cameras.
- It cannot generate new images of the moon based on raw data but will only work with pre-captured input images.
- The simulation is limited to a static model and does not include dynamic changes (e.g., changing lunar phases or real-time environmental variations on the moon's surface).

3.5 TECHNOLOGY

The project uses the following technologies:

- PyQt5: For creating the graphical user interface (GUI) that allows users to input and manipulate parameters.
- PyQtGraph: Used for visualization, it provides a fast and efficient way to display simulated images and graphical representations of camera settings.
- GDAL: A library used to handle geographic data and transform spatial data for accurate simulation of images.
- NumPy: Essential for performing mathematical operations and data transformations involved in the image simulation.

3.6 PROJECT PLANNING

3.6.1 Project Development Approach and Justification

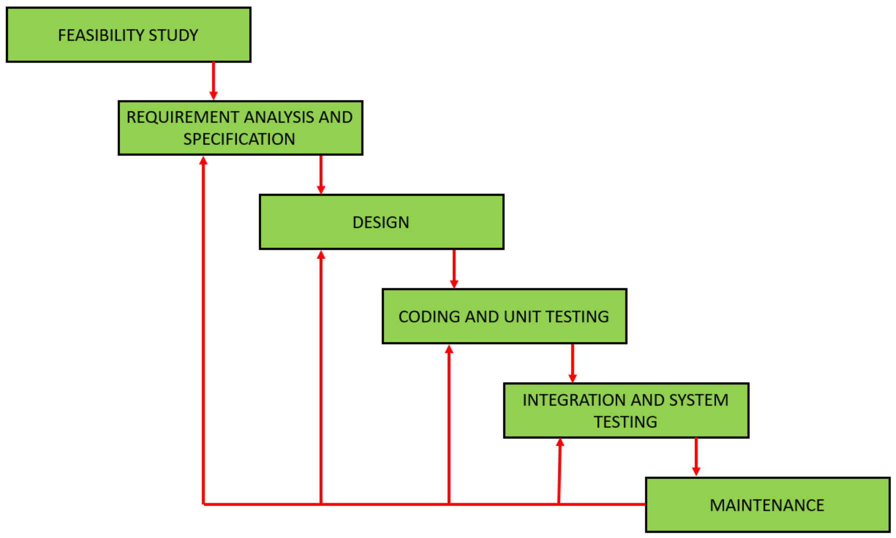

FEASIBILITY STUDY
REQUIREMENT ANALYSIS AND SPECIFICATION
DESIGN
CODING AND UNIT TESTING
INTEGRATION AND SYSTEM TESTING
MAINTENANCE

Figure 3.1: Iterative Waterfall Model

This project follows the Iterative Waterfall Model, allowing for incremental development and continuous feedback after each phase. The iterative model provides flexibility to refine the design after each cycle, ensuring that user needs are met and issues are addressed promptly.

Phase 1: Requirements Gathering and Initial Design: In this phase, project requirements were gathered from experts in space imaging.

Phase 2: Prototype Development and User Interface Design: Based on the requirements, an initial prototype of the GUI was developed using PyQt5. Early simulations with predefined parameters were conducted to verify the system's functionality.

Phase 3: Iteration and Refinement: After initial feedback from users, the system was iteratively refined. New features like parameter adjustments and image enhancement tools were added based on user input.

3.6.2 Project Effort and Time, Cost Estimation

The project is expected to take 3-4 months for completion. This includes time for prototype development, testing, and refinement of the system.

Time Estimation:

- Prototype development: 1 month
- Testing and refinement: 2 months

- Final adjustments and presentation: 1 month

Cost Estimation: The project will require already available hardware for testing.

4 SYSTEM ANALYSIS

4.1 STUDY OF CURRENT SYSTEM

Currently, there is no integrated system that allows users to simulate the appearance of the moon's surface based on varying camera parameters in a user-friendly way. Various image processing and satellite simulation tools exist but often focus on post-processing images or limited camera settings. Therefore, a need exists for a customizable, efficient tool that can provide simulations of lunar surface images with the flexibility to adjust camera positions, orientations, and other parameters.

Existing systems for space imaging simulations may lack features like interactive user inputs or adjustments for environmental factors such as altitude, yaw, and pitch. They also tend to be rigid in their design, focusing mainly on theoretical models rather than offering detailed user-driven simulations.

4.2 REQUIREMENTS OF NEW SYSTEM

The new system must fulfill the following requirements:

1. Customizable Parameters: The system should allow users to input and modify camera parameters, such as altitude, latitude, longitude, pitch, yaw, and roll.
2. User-Friendly Interface: The system must have an intuitive graphical user interface (GUI) developed using PyQt5, enabling easy input of parameters and viewing of simulated results.
3. Realistic Simulations: It should simulate realistic moon surface images using pre-captured satellite images.
4. Support for Lunar Mission Data: The system must allow integration with data from lunar missions such as Chandrayaan, enabling the simulation of different lunar regions and testing of different camera configurations.

4.3 SYSTEM FEASIBILITY

4.3.1 Does the system contribute to the overall objectives of the organization?

Yes, the system contributes to the overall objectives of ISRO, especially in the area of satellite imaging and space exploration. By helping engineers design and set up optimal camera parameters for lunar missions, the tool supports space mission readiness and camera configuration optimization. It aligns with the vision of advancing space technology and enhancing satellite-based remote sensing for lunar exploration.

4.3.2 Can the system be implemented using the current technology and schedule constraints?

Yes, the system can be implemented using current technologies such as PyQt5, NumPy, PyQtGraph, and GDAL. These tools are highly capable for developing the GUI, performing mathematical computations, and processing satellite image data. Additionally, the iterative waterfall model allows for efficient development within the time constraints, as each phase will build upon feedback from the previous iteration.

4.3.3 Can the system be integrated with other systems already in place?

Yes, the system is designed to integrate with existing simulation modules

4.4 PROCESS IN PROPOSED SYSTEM

The proposed system will include the following activities:

- Input Data: Users will provide pre-captured lunar images and set parameters such as latitude, longitude, altitude, camera orientation, etc.
- Image Simulation: Based on the input parameters, the system will simulate a new image of the moon's surface, reflecting how the camera would capture the scene in those specific parameters.
- Visualization: Simulated images will be visualized on a graphical interface, and users can interact with the interface to adjust parameters.
- Export: After running the simulation, users can export the generated images or data for further analysis.

4.5 FEATURES OF PROPOSED SYSTEM

- Interactive User Interface: Developed using PyQt5, allowing users to easily input and modify camera parameters and view simulated images.
- Simulation: Based on pre-defined camera parameters and the moon's surface data.
- Parameter Control: Includes controls for various camera settings (e.g., latitude, longitude, altitude, pitch, yaw, roll).
- Export and Report Generation: Users can save simulated images and generate detailed reports for mission planning.

4.6 LIST OF MAIN MODULES / COMPONENTS / PROCESSES / TECHNIQUES OF NEW SYSTEM /PROPOSED SYSTEM

The main modules/components of the system will include:

- User Interface Module: Developed with PyQt5 for user input and interaction.
- Simulation: Handles image processing and simulation based on camera parameters using NumPy and GDAL.
- Visualization Module: Uses PyQtGraph to display the simulated image and allow interactive viewing.
- Data Integration Module: Interfaces with satellite data for real-time simulation.
- Export/Reporting Module: Allows users to export simulated images for analysis.

4.7 SELECTION OF HARDWARE / SOFTWARE / ALGORITHMS / METHODOLOGY /TECHNIQUES

4.7.1 Hardware:

The project will be developed on standard computers with sufficient computational power to handle image processing. Systems should have good graphic processing capabilities for efficient simulation and rendering.

4.7.2 Software:

- PyQt5 for GUI development
- NumPy for mathematical operations
- PyQtGraph for image visualization
- GDAL for handling geospatial data

4.7.3 Algorithms:

Parallel processing algorithms were implemented for fast processing.

4.7.4 Methodology:

The project will follow the Iterative Waterfall Model, ensuring that each phase undergoes continuous refinement through feedback loops.

4.7.5 Techniques:

The system will employ camera parameter optimization and image simulation techniques to provide users with a realistic simulation of lunar surface imagery.

5 SYSTEM DESIGN

5.1 SYSTEM DESIGN & METHODOLOGY

The image simulation system is designed to take a GeoTIFF image as input and simulate how a camera would capture it in an unknown environment. The methodology follows a structured pipeline consisting of image loading, parameter configuration, marker placement, and simulation output. it's made through Iterative Waterfall Model.

The key components of the system design are:

5.1.1 Graphical User Interface (GUI)

Built using PyQt and PyQtGraph, providing an interactive environment for parameter adjustments and image visualization.

5.1.2 Camera Parameters Handling

Users can set parameters such as detector height, width, FOV, focal length, and altitude to define the camera specifications.

5.1.3 Marker-Based Simulation

Users can place markers using latitude-longitude coordinates to define the simulation area. Single-point mode allows one marker, while multi-point mode enables simulation between two points.

5.1.4 Footprint Preview

The system provides a footprint visualization before actual simulation to help users understand the expected output coverage.

5.1.5 Image Processing

The core simulation logic processes the input GeoTIFF based on selected camera parameters, generating the simulated image.

This structured approach ensures that the system effectively meets the objective of simulating camera behavior for testing and design purposes.

5.2 DATA STRUCTURE /PROCESS /STRUCTURE DESIGN

5.2.1 Data Structure Design

The system primarily works with the following data structures:

- Camera Parameter Dictionary: Stores values such as detector size, FOV, focal length, and altitude.
- GeoTIFF Image Data: Handled as multi-dimensional arrays for processing.
- Marker Data: Stores latitude-longitude pairs for single and multi-point modes.

5.2.2 Process Design

The image simulation follows these steps:

1. Load Image: User selects a GeoTIFF image.
2. Set Camera Parameters: Parameters are either manually entered or selected from the preloaded camera types via a combo box.
3. Mark Position(s): User marks a single or two-point location using the mouse or by entering coordinates.
4. Preview Footprint: The system calculates and displays the expected coverage.
5. Simulate Image: The system processes the image according to camera parameters and outputs the simulated image.
6. Save or Clear: Users can save results or reset parameters and markers.

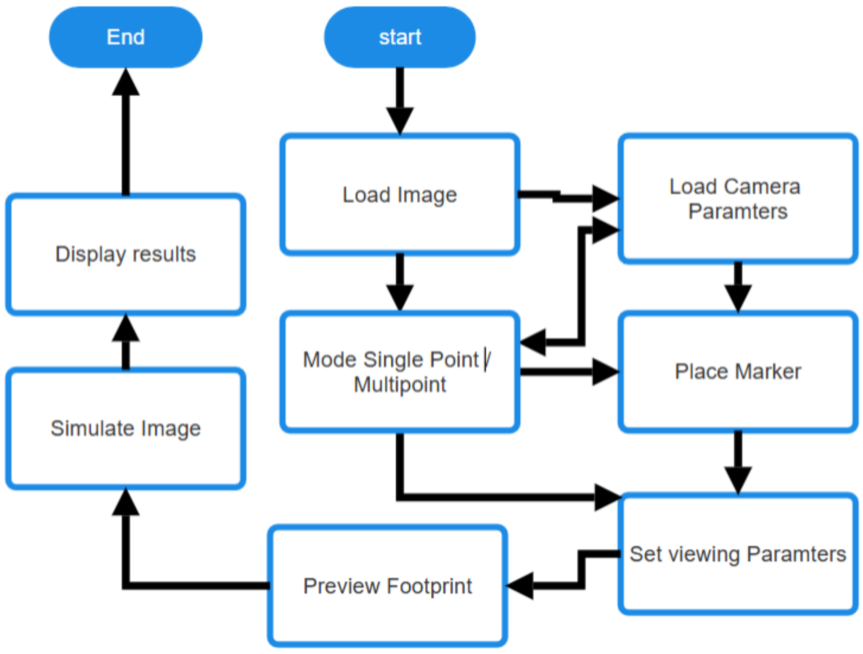


Figure 5.1: User Flow Chart

5.3 INPUT / OUTPUT AND INTERFACE DESIGN

5.3.1 State Transition

State transition depict the various states of the system,

such as:

Idle → Image Loaded → Parameters Set → Simulation Performed → Output Displayed

5.3.1 Samples of Forms, Reports, and Interface

- Forms & Inputs: The GUI provides QLineEdit fields for numerical input and a QComboBox for selecting preloaded camera types.
- Interactive Controls: Buttons for loading, saving, and clearing data, along with QDials for adjusting roll, pitch, yaw, and tilt angles.
- Visualization: The PyQtGraph canvas displays the input and output images with an interactive histogram.

6 IMPLEMENTATION

6.1 IMPLEMENTATION PLATFORM /ENVIRONMENT

The image simulation system is developed using Python and implemented on a Windows/Linux environment. The key components of the implementation platform are:

- Programming Language: Python
- GUI Framework: PyQt5 (for creating an interactive graphical interface)
- Image Processing Library: GDAL (for handling GeoTIFF images)
- Visualization Library: PyQtGraph (for displaying images and histograms)
- Mathematical Computation: NumPy (for handling array-based image transformations)
- Development Environment: Spyder
- Hardware Requirements: A system with a minimum of 8GB RAM, multi-core CPU, and GPU acceleration

The system runs efficiently on standard computing hardware, with optimizations to handle large GeoTIFF files.

6.2 PROCESS / PROGRAM / TECHNOLOGY / MODULES SPECIFICATION(S)

6.2.1 Process Flow

The image simulation follows these steps:

1. Loading the Input Image:
    - Users load a GeoTIFF file using a file selection dialog.

- The image data is read using the GDAL library and processed into an array for simulation.

2. Setting Camera Parameters:
   - Users manually enter or select a predefined camera type from a QComboBox.
   - Parameters such as detector size, focal length, field of view, and altitude are applied.
3. Marking Positions (Single or Multi-Point Mode):
   - Users click on the image or enter latitude-longitude coordinates to set markers.
   - The system converts these to pixel coordinates for processing.
4. Footprint Visualization:
   - A footprint button calculates and overlays the expected simulated area on the original image.
5. Simulating the Image:
   - The system applies geometric transformations based on camera parameters.
   - The modified image is displayed on the PyQtGraph canvas.
6. Saving / Resetting Data:
   - Users can save results, reset inputs, or clear markers as needed.

6.2.2 Technologies & Modules Used

The following technologies and modules are used in the implementation:

- PyQt5: GUI framework for interactive controls.
- PyQtGraph: Efficient image rendering and histogram display.
- GDAL: Reads and processes GeoTIFF images.
- NumPy: Handles mathematical transformations for image simulation.

6.2.3 Findings / Results / Outcomes

The implementation of the system resulted in the following outcomes:

- Accurate Image Simulation: The system successfully simulates how a camera would capture an image in an unknown environment.

- Parameter Adjustment: Users can modify camera parameters and observe changes dynamically.
- Footprint Visualization: The footprint feature helps users understand how the final image will appear before simulation.
- Multi-Mode Simulation: Both single-point and multi-point simulation modes function as expected.
- Efficient Processing: The use of NumPy and GDAL ensures that large GeoTIFF images are processed efficiently.

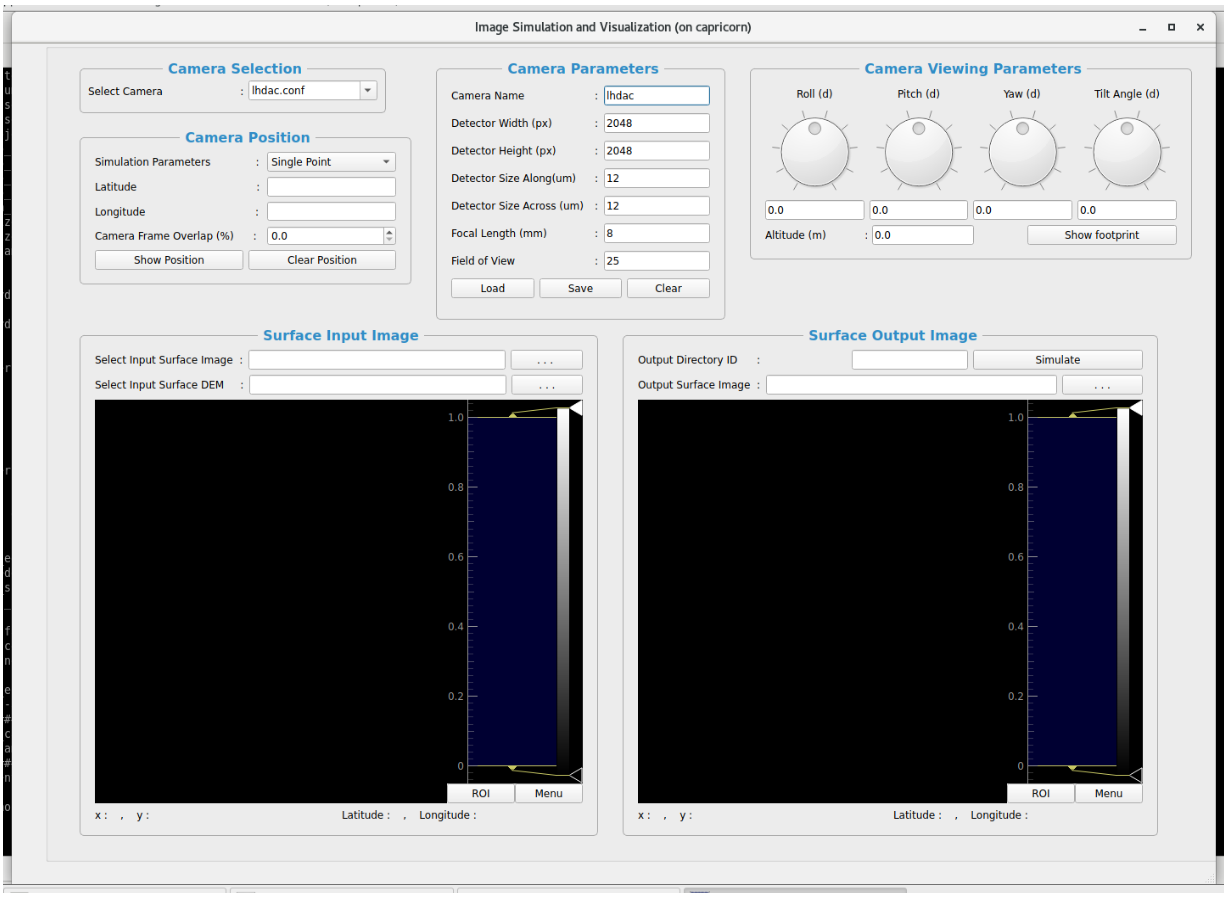
6.2.4 Result Analysis / Comparison / Deliberations

The system was tested with various input images and camera parameters to evaluate its performance. The key observations include:

- Comparison with Theoretical Expectations: The simulated images align with expected outputs based on camera equations.
- Performance Metrics: Processing time varies based on image resolution and selected parameters. Higher-resolution images take longer to process.
- User Interaction: The GUI provides an intuitive interface for parameter adjustments, making the system user-friendly.
- Challenges: Some minor challenges were faced in marker precision, which were mitigated using coordinate transformations.

6.2.5 Implementation :

- Screenshots :



Select Camera : lhdac.conf

Simulation Parameters : Single Point
Latitude :
Longitude :
Camera Frame Overlap (%) : 0.0

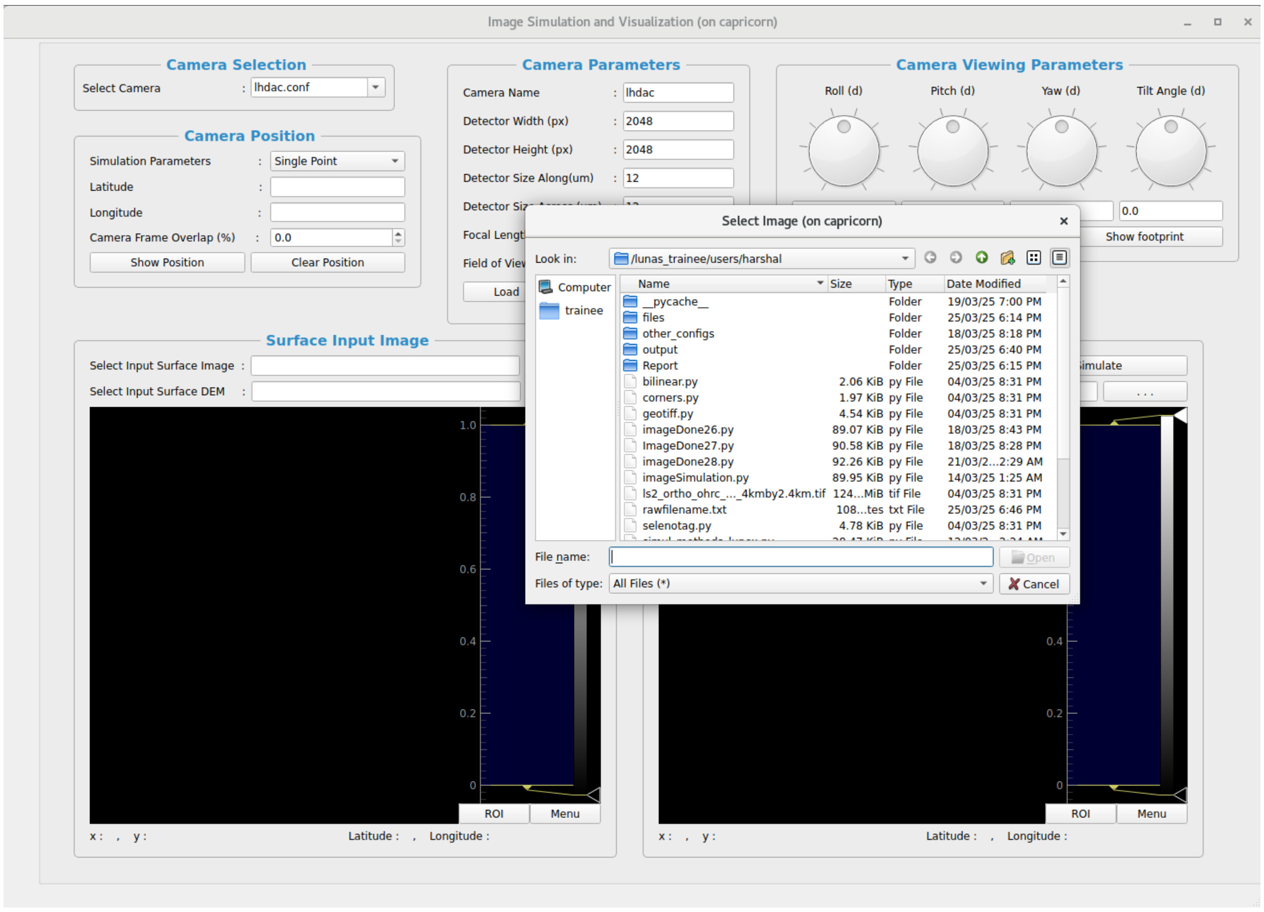
Camera Name : lhdac
Detector Width (px) : 2048
Detector Height (px) : 2048
Detector Size Along(um) : 12
Detector Size Across (um) : 12
Focal Length (mm) : 8
Field of View : 25

Roll (d) 0.0 Pitch (d) 0.0 Yaw (d) 0.0 Tilt Angle (d) 0.0
Altitude (m) : 0.0

Select Input Surface Image :
Select Input Surface DEM :

Output Directory ID :
Output Surface Image :

Figure 6.1 Launching UI



Select Camera : lhdac.conf

Simulation Parameters : Single Point
Latitude :
Longitude :
Camera Frame Overlap (%) : 0.0

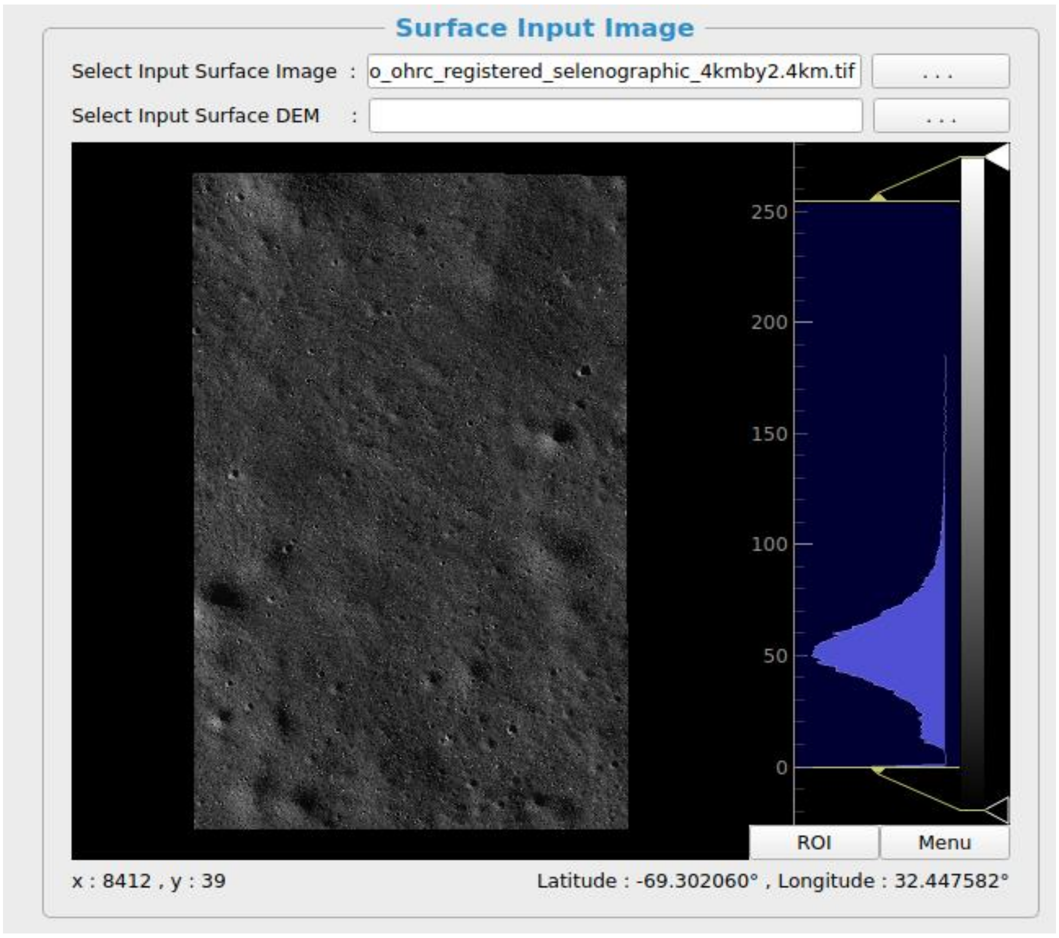
Camera Name : lhdac
Detector Width (px) : 2048
Detector Height (px) : 2048
Detector Size Along(um) : 12

Look in: /lunas_trainee/users/harshal

Name	Size	Type	Date Modified
__pycache__		Folder	19/03/25 7:00 PM
files		Folder	25/03/25 6:14 PM
other_configs		Folder	18/03/25 8:18 PM
output		Folder	25/03/25 6:40 PM
Report		Folder	25/03/25 6:15 PM
bilinear.py	2.06 KiB	py File	04/03/25 8:31 PM
corners.py	1.97 KiB	py File	04/03/25 8:31 PM
geotiff.py	4.54 KiB	py File	04/03/25 8:31 PM
imageDone26.py	89.07 KiB	py File	18/03/25 8:43 PM
ImageDone27.py	90.58 KiB	py File	18/03/25 8:28 PM
imageDone28.py	92.26 KiB	py File	21/03/2...2:29 AM
imageSimulation.py	89.95 KiB	py File	14/03/25 1:25 AM
ls2_ortho_ohrc_..._4kmby2.4km.tif	124...MiB	tif File	04/03/25 8:31 PM
rawfilename.txt	108...tes	txt File	25/03/25 6:46 PM
selenotag.py	4.78 KiB	py File	04/03/25 8:31 PM

File name:
Files of type: All Files (*)

Figure 6.2 Loading a GeoTif Image


Surface Input Image

Select Input Surface Image : o_ohrc_registered_selenographic_4kmby2.4km.tif

Select Input Surface DEM :

x : 8412 , y : 39

Latitude : -69.302060° , Longitude : 32.447582°

Figure 6.3 Surface Input image

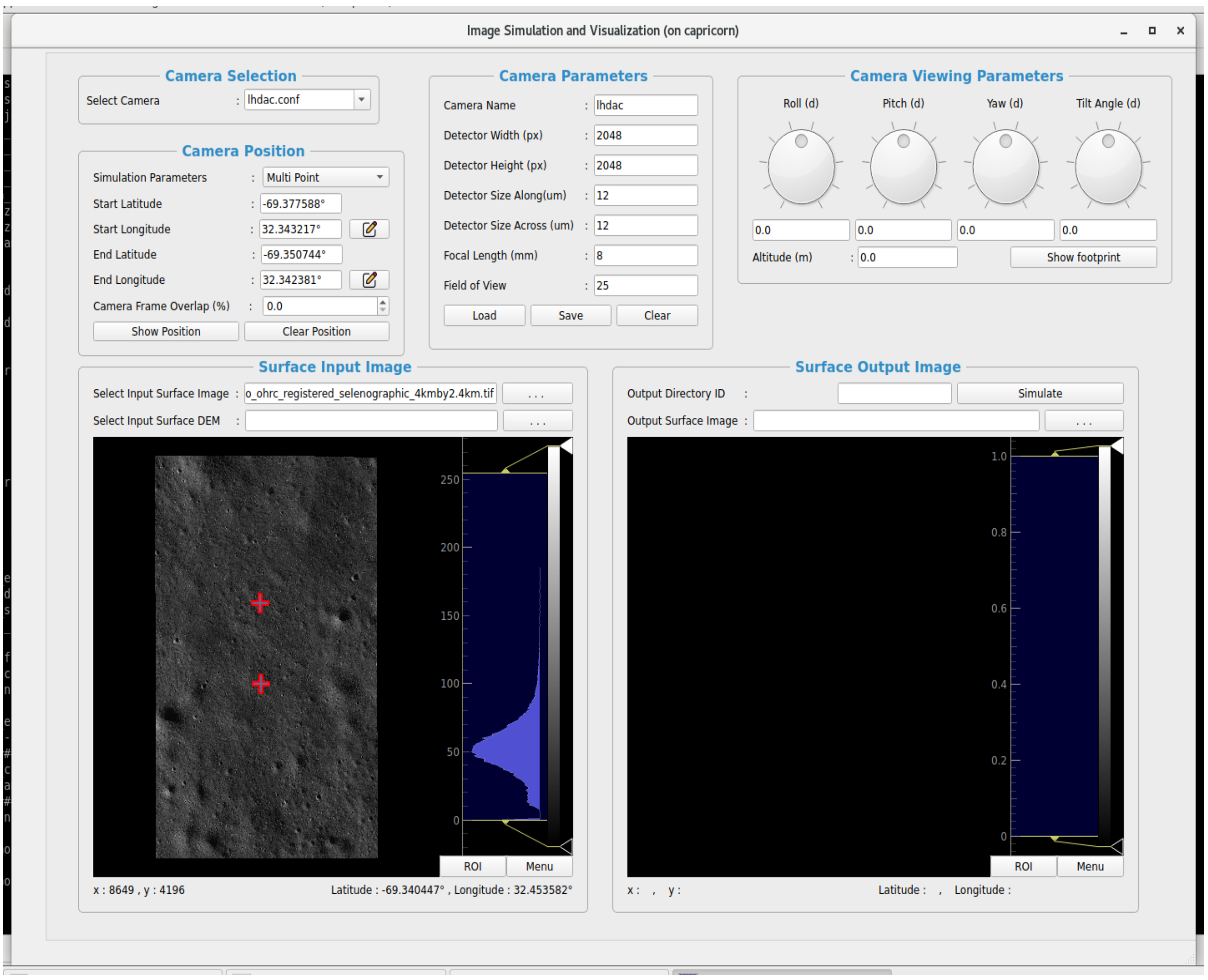


Image Simulation and Visualization (on capricorn)

Camera Selection

Select Camera : lhdac.conf

Camera Position

Simulation Parameters	Multi Point
Start Latitude	-69.377588°
Start Longitude	32.343217°
End Latitude	-69.350744°
End Longitude	32.342381°
Camera Frame Overlap (%)	0.0

Camera Parameters

Camera Name	lhdac
Detector Width (px)	2048
Detector Height (px)	2048
Detector Size Along(um)	12
Detector Size Across (um)	12
Focal Length (mm)	8
Field of View	25

Camera Viewing Parameters

Roll (d)	Pitch (d)	Yaw (d)	Tilt Angle (d)
0.0	0.0	0.0	0.0

Altitude (m) : 0.0

Surface Input Image

Select Input Surface Image : o_ohrc_registered_selenographic_4kmby2.4km.tif

Select Input Surface DEM :

x : 8649 , y : 4196

Latitude : -69.340447° , Longitude : 32.453582°

Surface Output Image

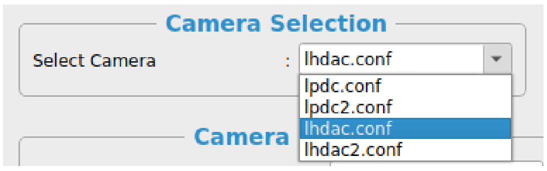
Output Directory ID :

Output Surface Image :

x : , y :

Latitude : , Longitude :

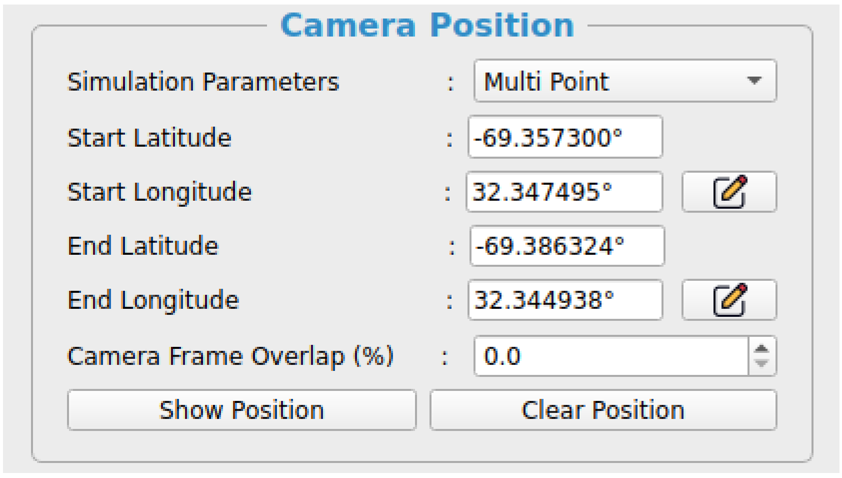
Figure 6.4 Visualization of Multipoint


Camera Selection

Select Camera : lhdac.conf

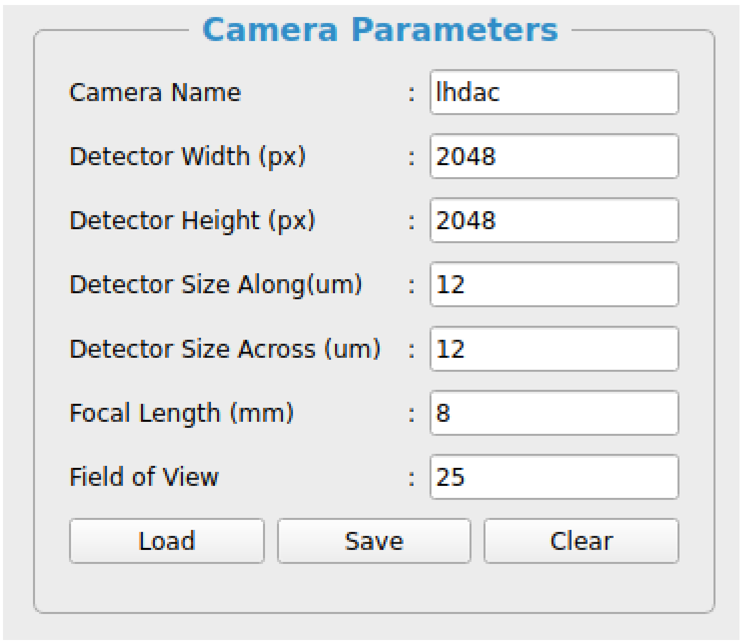
- lpdc.conf
- lpdc2.conf
- lhdac.conf
- lhdac2.conf

Figure 6.5 Camera Selection


Camera Position

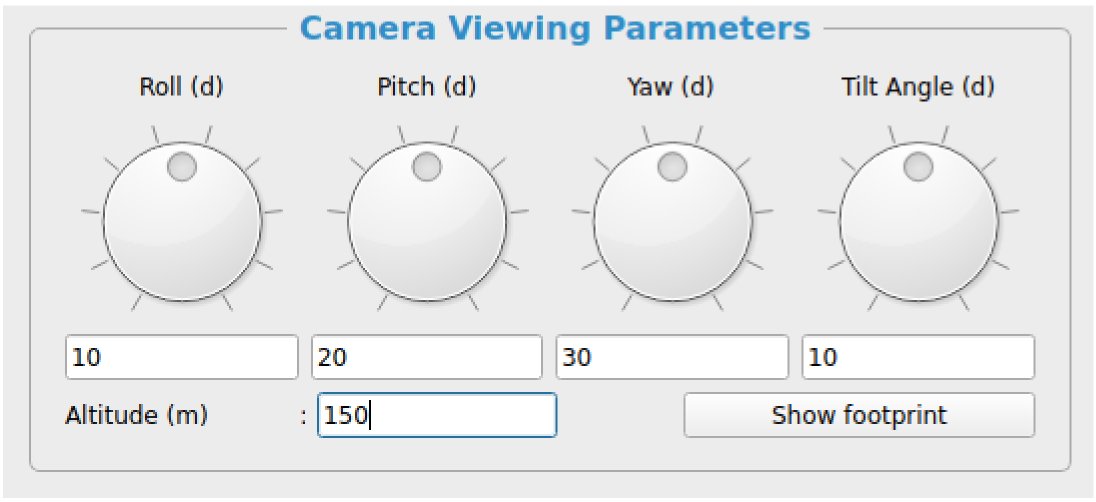
Simulation Parameters	:	Multi Point
Start Latitude	:	-69.357300°
Start Longitude	:	32.347495°
End Latitude	:	-69.386324°
End Longitude	:	32.344938°
Camera Frame Overlap (%)	:	0.0

Figure 6.6 Multipoint Camera Selection


Camera Parameters

Camera Name	:	lhdac
Detector Width (px)	:	2048
Detector Height (px)	:	2048
Detector Size Along(um)	:	12
Detector Size Across (um)	:	12
Focal Length (mm)	:	8
Field of View	:	25

Figure 6.7 Camera Parameters


Camera Viewing Parameters

Roll (d)	Pitch (d)	Yaw (d)	Tilt Angle (d)
10	20	30	10

Altitude (m) : 150

Figure 6.8 Camera Viewing Parameters

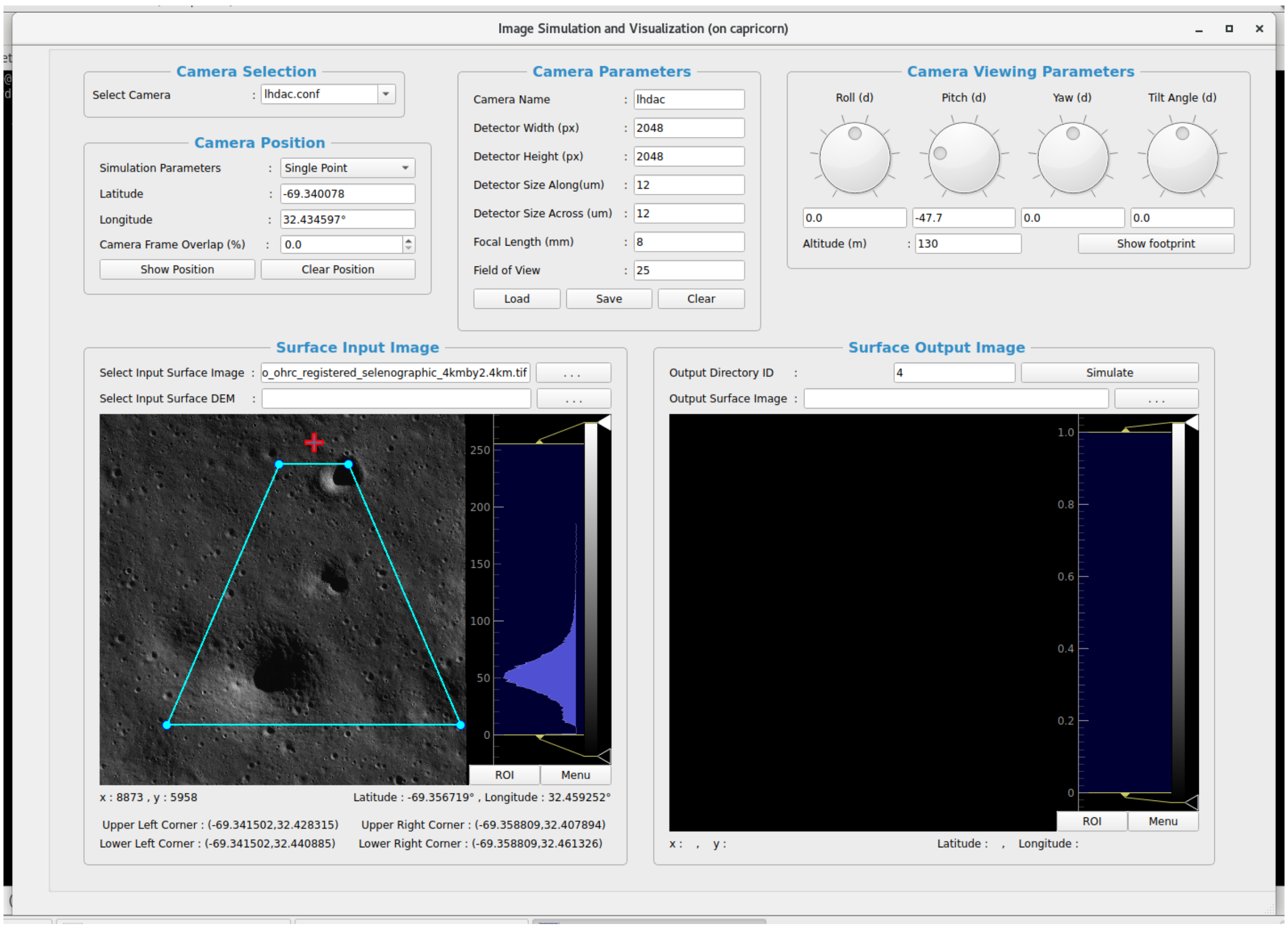


Image Simulation and Visualization (on capricorn)

Camera Selection

Select Camera : lhdac.conf

Camera Position

Simulation Parameters : Single Point
Latitude : -69.340078
Longitude : 32.434597°
Camera Frame Overlap (%) : 0.0

Camera Parameters

Camera Name : lhdac
Detector Width (px) : 2048
Detector Height (px) : 2048
Detector Size Along(um) : 12
Detector Size Across (um) : 12
Focal Length (mm) : 8
Field of View : 25

Camera Viewing Parameters

Roll (d): 0.0
Pitch (d): -47.7
Yaw (d): 0.0
Tilt Angle (d): 0.0
Altitude (m) : 130

Surface Input Image

Select Input Surface Image : o_ohrc_registered_selenographic_4kmby2.4km.tif
Select Input Surface DEM :

x : 8873 , y : 5958   Latitude : -69.356719° , Longitude : 32.459252°

Upper Left Corner : (-69.341502,32.428315)   Upper Right Corner : (-69.358809,32.407894)
Lower Left Corner : (-69.341502,32.440885)   Lower Right Corner : (-69.358809,32.461326)

Surface Output Image

Output Directory ID : 4
Output Surface Image :

x : , y :   Latitude : , Longitude :

Figure 6.9 Computing Footprint

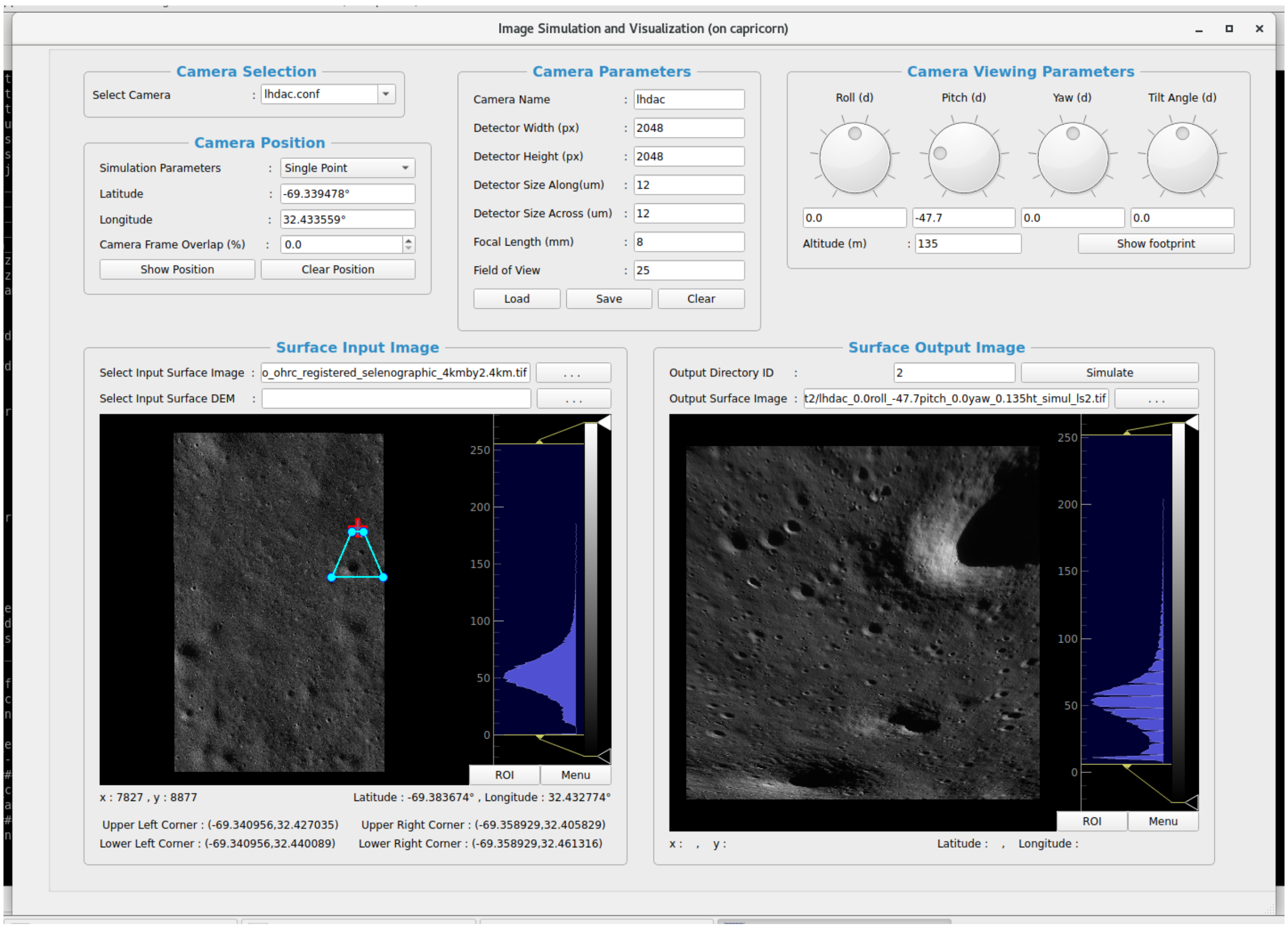


Image Simulation and Visualization (on capricorn)

Camera Selection

Select Camera : lhdac.conf

Camera Position

Simulation Parameters : Single Point
Latitude : -69.339478°
Longitude : 32.433559°
Camera Frame Overlap (%) : 0.0

Camera Parameters

Camera Name : lhdac
Detector Width (px) : 2048
Detector Height (px) : 2048
Detector Size Along(um) : 12
Detector Size Across (um) : 12
Focal Length (mm) : 8
Field of View : 25

Camera Viewing Parameters

Roll (d): 0.0
Pitch (d): -47.7
Yaw (d): 0.0
Tilt Angle (d): 0.0
Altitude (m) : 135

Surface Input Image

Select Input Surface Image : o_ohrc_registered_selenographic_4kmby2.4km.tif
Select Input Surface DEM :

x : 7827 , y : 8877   Latitude : -69.383674° , Longitude : 32.432774°

Upper Left Corner : (-69.340956,32.427035)   Upper Right Corner : (-69.358929,32.405829)
Lower Left Corner : (-69.340956,32.440089)   Lower Right Corner : (-69.358929,32.461316)

Surface Output Image

Output Directory ID : 2
Output Surface Image : t2/lhdac_0.0roll_-47.7pitch_0.0yaw_0.135ht_simul_ls2.tif

x : , y :   Latitude : , Longitude :

Figure 6.10 Zoom out input image

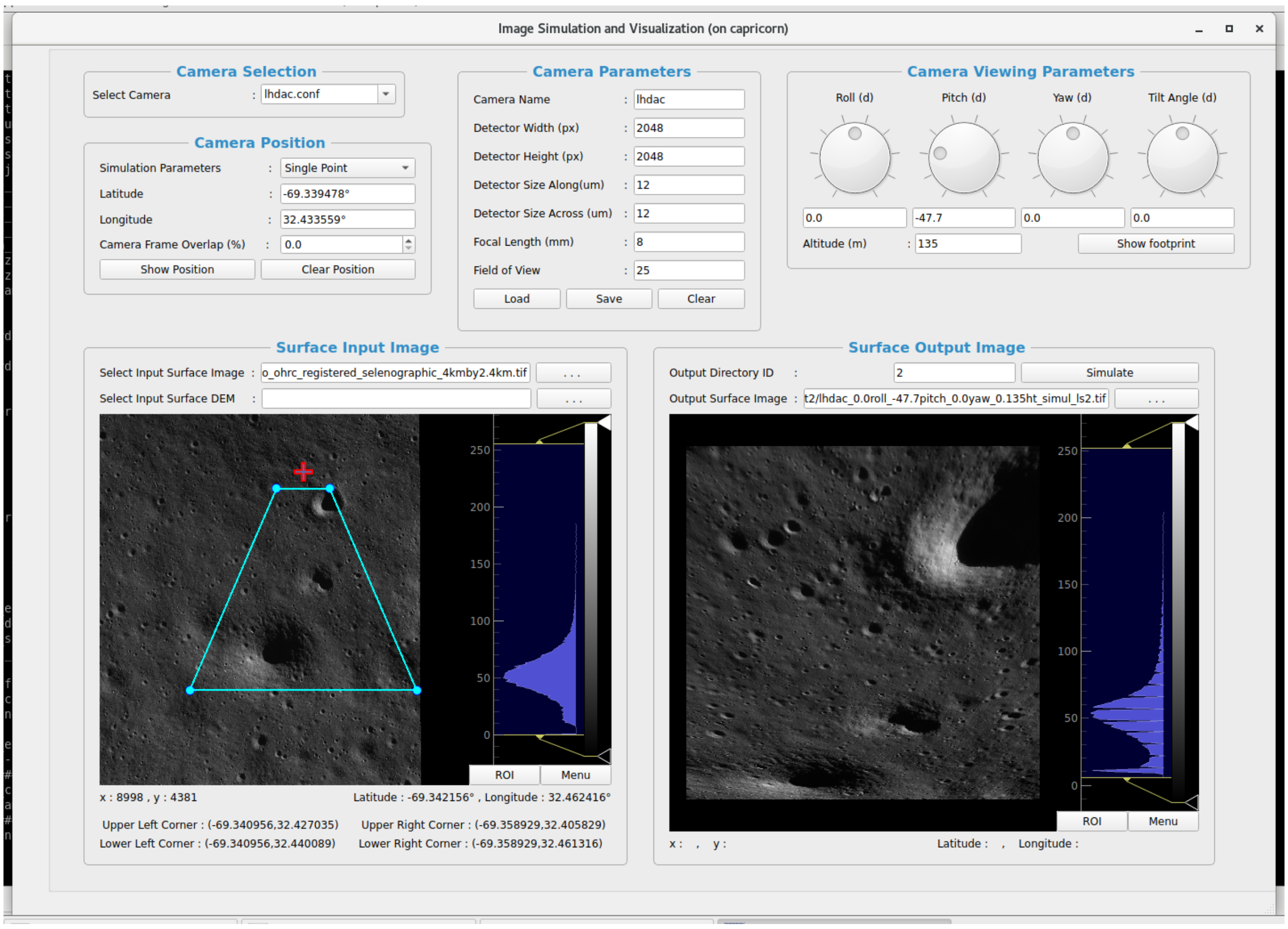


Image Simulation and Visualization (on capricorn)

Camera Selection

Select Camera : lhdac.conf

Camera Position

Simulation Parameters : Single Point

Latitude : -69.339478°

Longitude : 32.433559°

Camera Frame Overlap (%) : 0.0

Camera Parameters

Camera Name : lhdac

Detector Width (px) : 2048

Detector Height (px) : 2048

Detector Size Along(um) : 12

Detector Size Across (um) : 12

Focal Length (mm) : 8

Field of View : 25

Camera Viewing Parameters

Roll (d) 0.0 | Pitch (d) -47.7 | Yaw (d) 0.0 | Tilt Angle (d) 0.0

Altitude (m) : 135

Surface Input Image

Select Input Surface Image : o_ohrc_registered_selenographic_4kmby2.4km.tif

Select Input Surface DEM :

x : 8998 , y : 4381 Latitude : -69.342156° , Longitude : 32.462416°

Upper Left Corner : (-69.340956,32.427035) Upper Right Corner : (-69.358929,32.405829)

Lower Left Corner : (-69.340956,32.440089) Lower Right Corner : (-69.358929,32.461316)

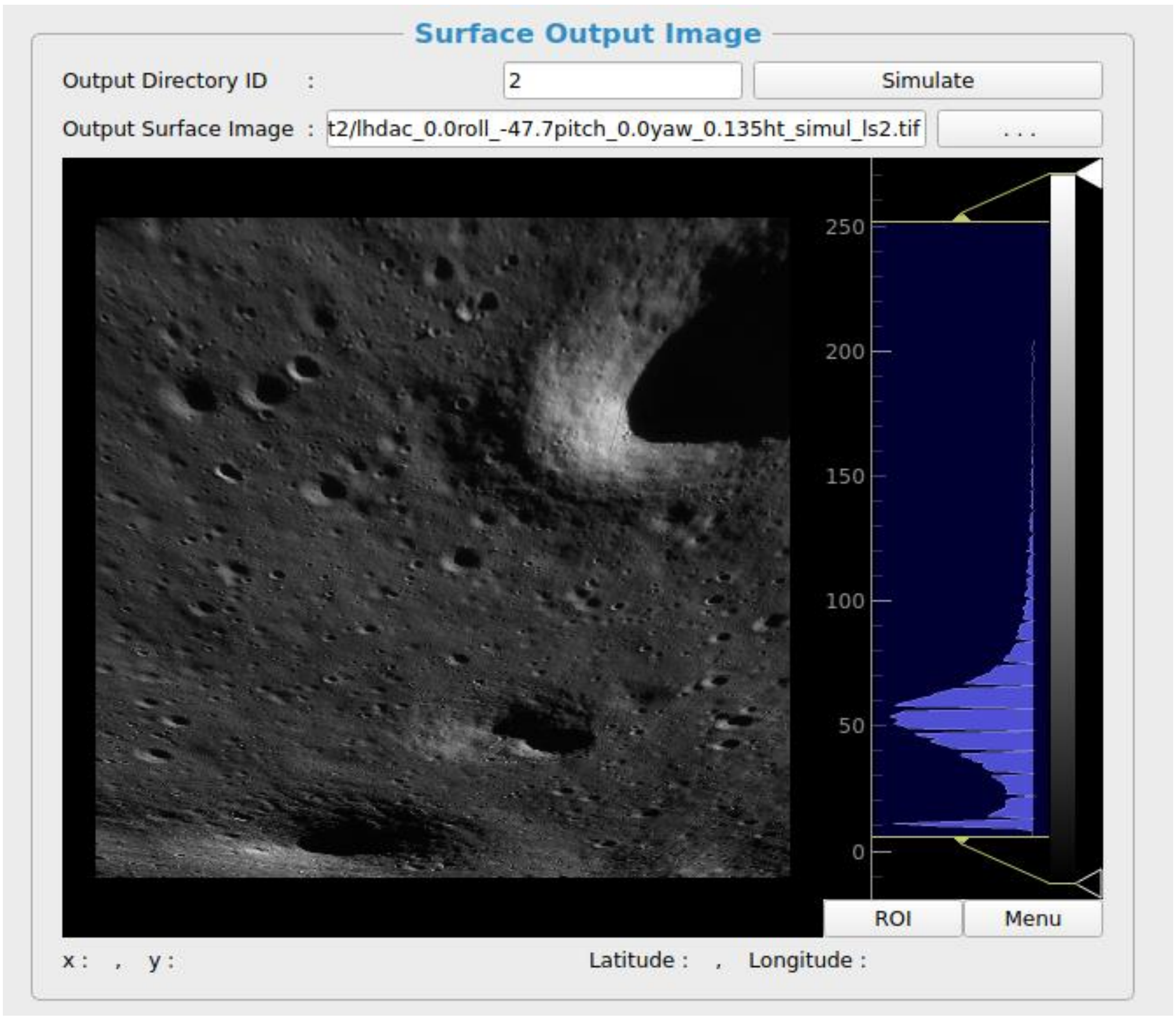
Surface Output Image

Output Directory ID : 2

Output Surface Image : t2/lhdac_0.0roll_-47.7pitch_0.0yaw_0.135ht_simul_ls2.tif

x : , y : Latitude : , Longitude :

Figure 6.11 Final Output


Surface Output Image

Output Directory ID : 2

Output Surface Image : t2/lhdac_0.0roll_-47.7pitch_0.0yaw_0.135ht_simul_ls2.tif

x : , y : Latitude : , Longitude :

Figure 6.12 Output Image

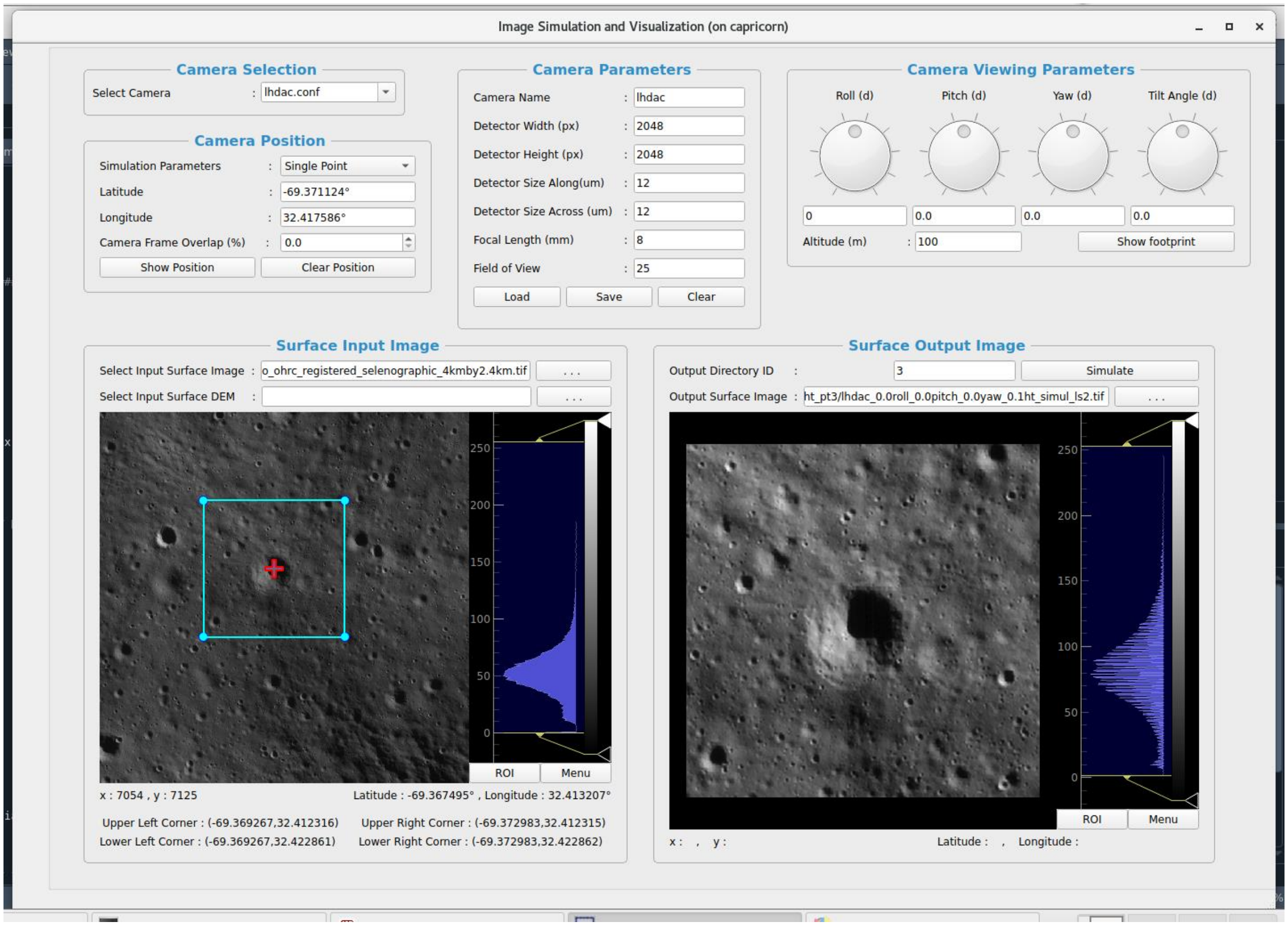


Image Simulation and Visualization (on capricorn)

Camera Selection

Select Camera : lhdac.conf

Camera Position

Simulation Parameters : Single Point

Latitude : -69.371124°

Longitude : 32.417586°

Camera Frame Overlap (%) : 0.0

Camera Parameters

Camera Name : lhdac

Detector Width (px) : 2048

Detector Height (px) : 2048

Detector Size Along(um) : 12

Detector Size Across (um) : 12

Focal Length (mm) : 8

Field of View : 25

Camera Viewing Parameters

Roll (d) : 0

Pitch (d) : 0.0

Yaw (d) : 0.0

Tilt Angle (d) : 0.0

Altitude (m) : 100

Surface Input Image

Select Input Surface Image : o_ohrc_registered_selenographic_4kmby2.4km.tif

Select Input Surface DEM :

x : 7054 , y : 7125 Latitude : -69.367495° , Longitude : 32.413207°

Upper Left Corner : (-69.369267,32.412316) Upper Right Corner : (-69.372983,32.412315)

Lower Left Corner : (-69.369267,32.422861) Lower Right Corner : (-69.372983,32.422862)

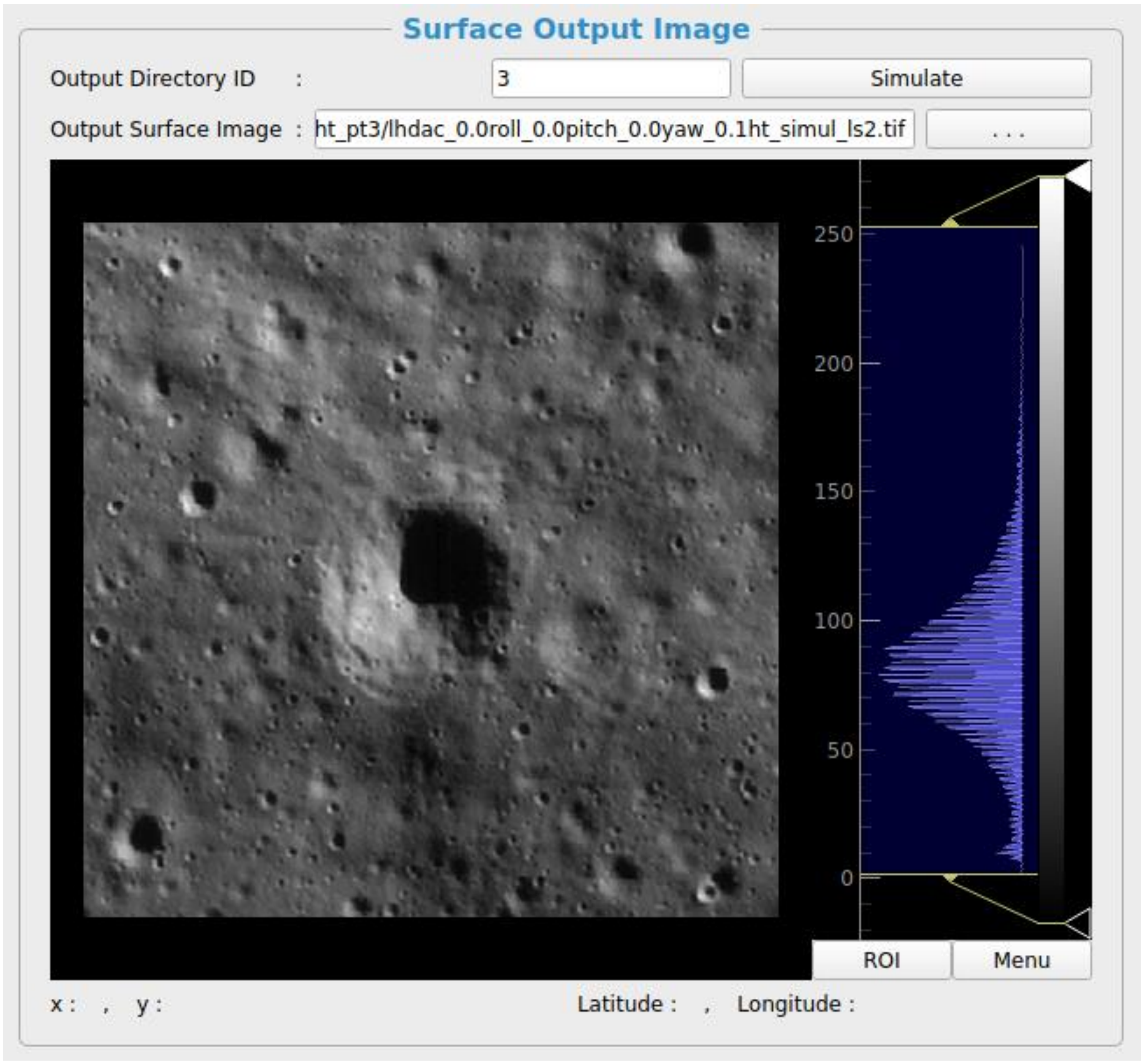
Surface Output Image

Output Directory ID : 3

Output Surface Image : ht_pt3/lhdac_0.0roll_0.0pitch_0.0yaw_0.1ht_simul_ls2.tif

x : , y : Latitude : , Longitude :

Figure 6.13 Example_1

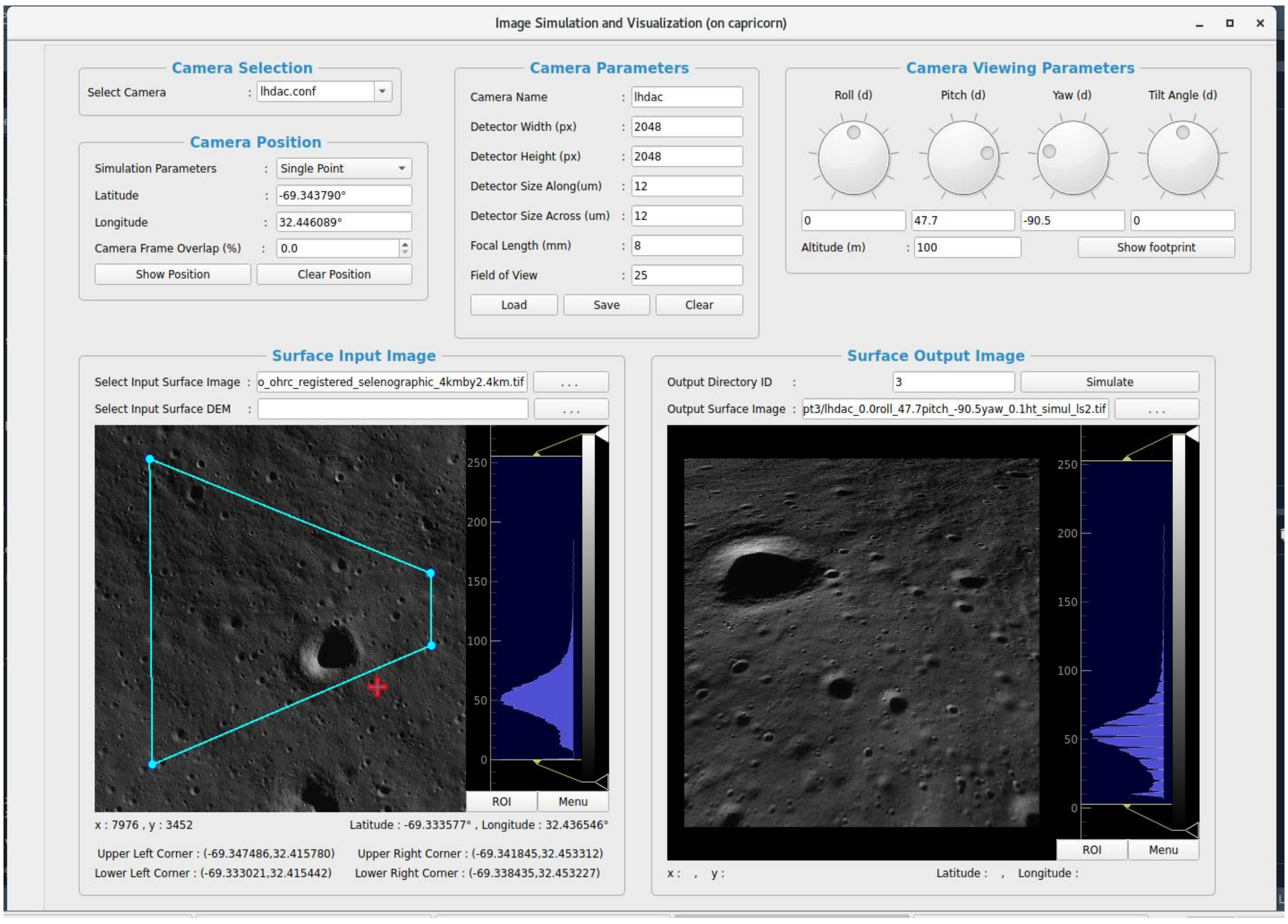

Surface Output Image

Output Directory ID : 3

Output Surface Image : ht_pt3/lhdac_0.0roll_0.0pitch_0.0yaw_0.1ht_simul_ls2.tif

x : , y : Latitude : , Longitude :

Figure 6.14 Example_1 Output



Camera Selection

Select Camera : lhdac.conf

Camera Position

Simulation Parameters : Single Point
Latitude : -69.343790°
Longitude : 32.446089°
Camera Frame Overlap (%) : 0.0

Camera Parameters

Camera Name : lhdac
Detector Width (px) : 2048
Detector Height (px) : 2048
Detector Size Along(um) : 12
Detector Size Across (um) : 12
Focal Length (mm) : 8
Field of View : 25

Camera Viewing Parameters

Roll (d) 0
Pitch (d) 47.7
Yaw (d) -90.5
Tilt Angle (d) 0
Altitude (m) : 100

Surface Input Image

Select Input Surface Image : o_ohrc_registered_selenographic_4kmby2.4km.tif
Select Input Surface DEM :

x : 7976 , y : 3452 Latitude : -69.333577° , Longitude : 32.436546°

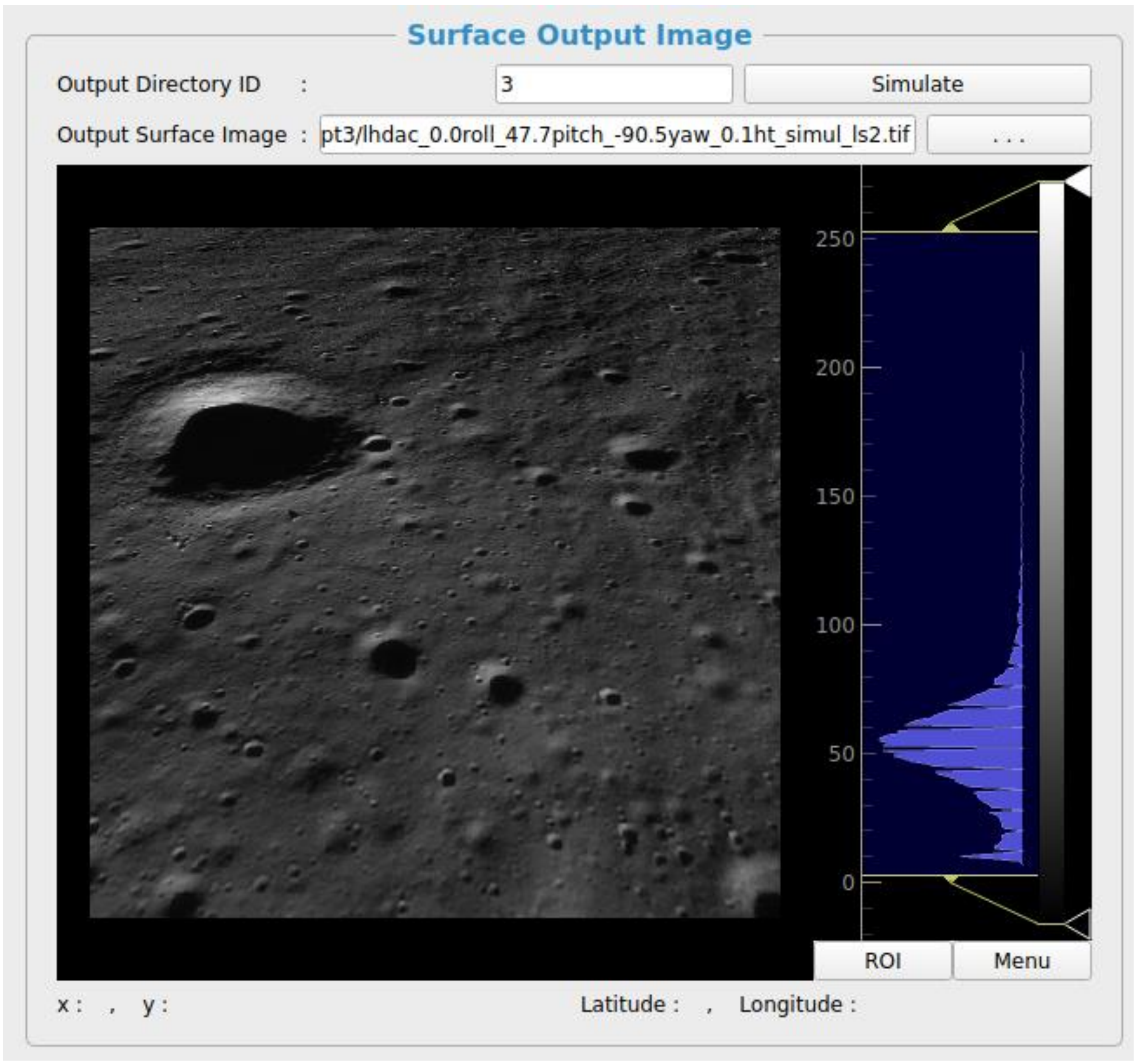
Upper Left Corner : (-69.347486,32.415780) Upper Right Corner : (-69.341845,32.453312)
Lower Left Corner : (-69.333021,32.415442) Lower Right Corner : (-69.338435,32.453227)

Surface Output Image

Output Directory ID : 3
Output Surface Image : pt3/lhdac_0.0roll_47.7pitch_-90.5yaw_0.1ht_simul_ls2.tif

x : , y : Latitude : , Longitude :

Figure 6.15 Example_2



Surface Output Image

Output Directory ID : 3
Output Surface Image : pt3/lhdac_0.0roll_47.7pitch_-90.5yaw_0.1ht_simul_ls2.tif

x : , y : Latitude : , Longitude :

Figure 6.16 Example_2 Output

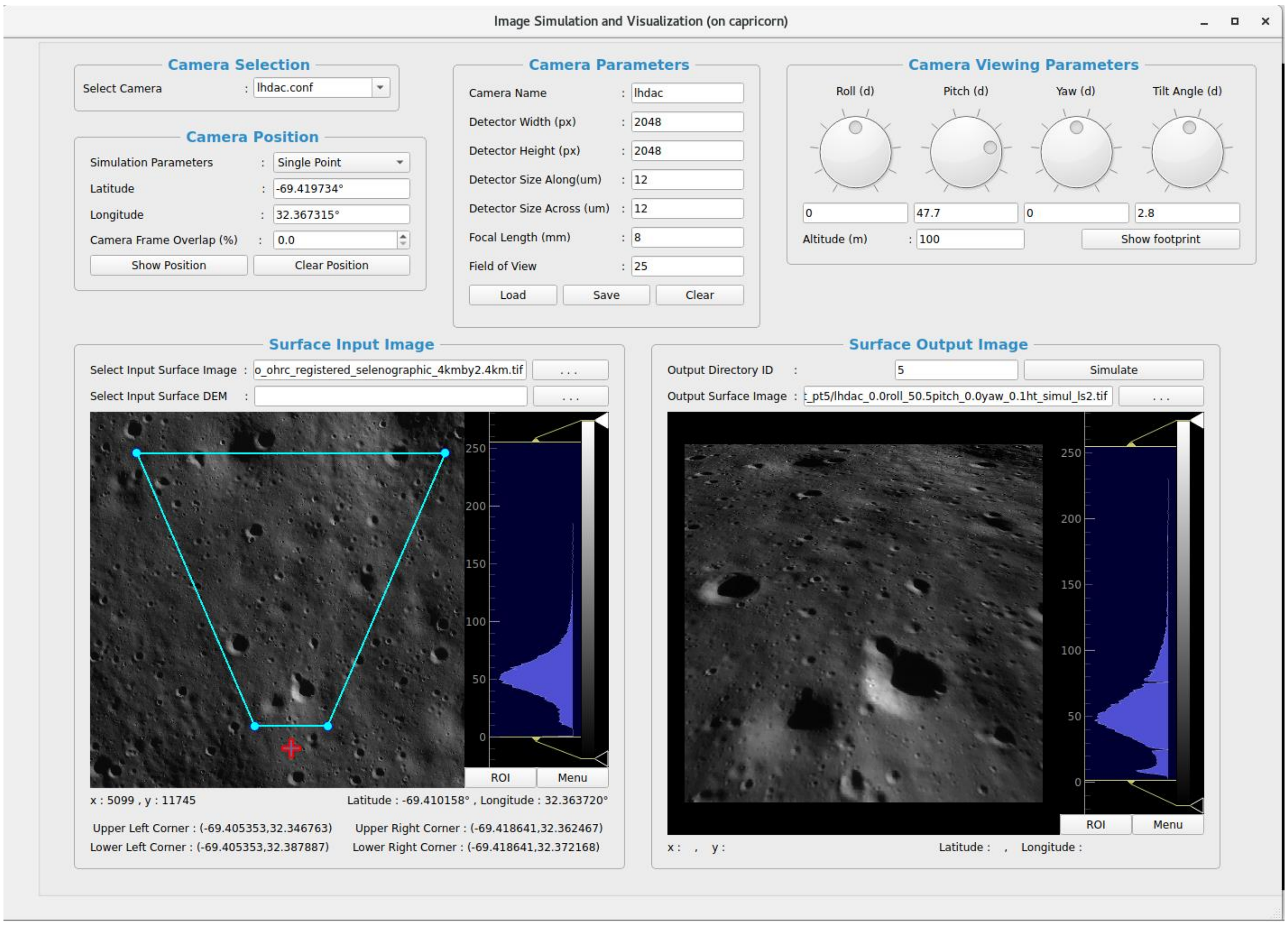


Image Simulation and Visualization (on capricorn)

Camera Selection

Select Camera : lhdac.conf

Camera Position

Simulation Parameters : Single Point
Latitude : -69.419734°
Longitude : 32.367315°
Camera Frame Overlap (%) : 0.0

Show Position | Clear Position

Camera Parameters

Camera Name : lhdac
Detector Width (px) : 2048
Detector Height (px) : 2048
Detector Size Along(um) : 12
Detector Size Across (um) : 12
Focal Length (mm) : 8
Field of View : 25

Load | Save | Clear

Camera Viewing Parameters

Roll (d)	Pitch (d)	Yaw (d)	Tilt Angle (d)
0	47.7	0	2.8

Altitude (m) : 100

Show footprint

Surface Input Image

Select Input Surface Image : o_ohrc_registered_selenographic_4kmby2.4km.tif
Select Input Surface DEM :

x : 5099 , y : 11745   Latitude : -69.410158° , Longitude : 32.363720°

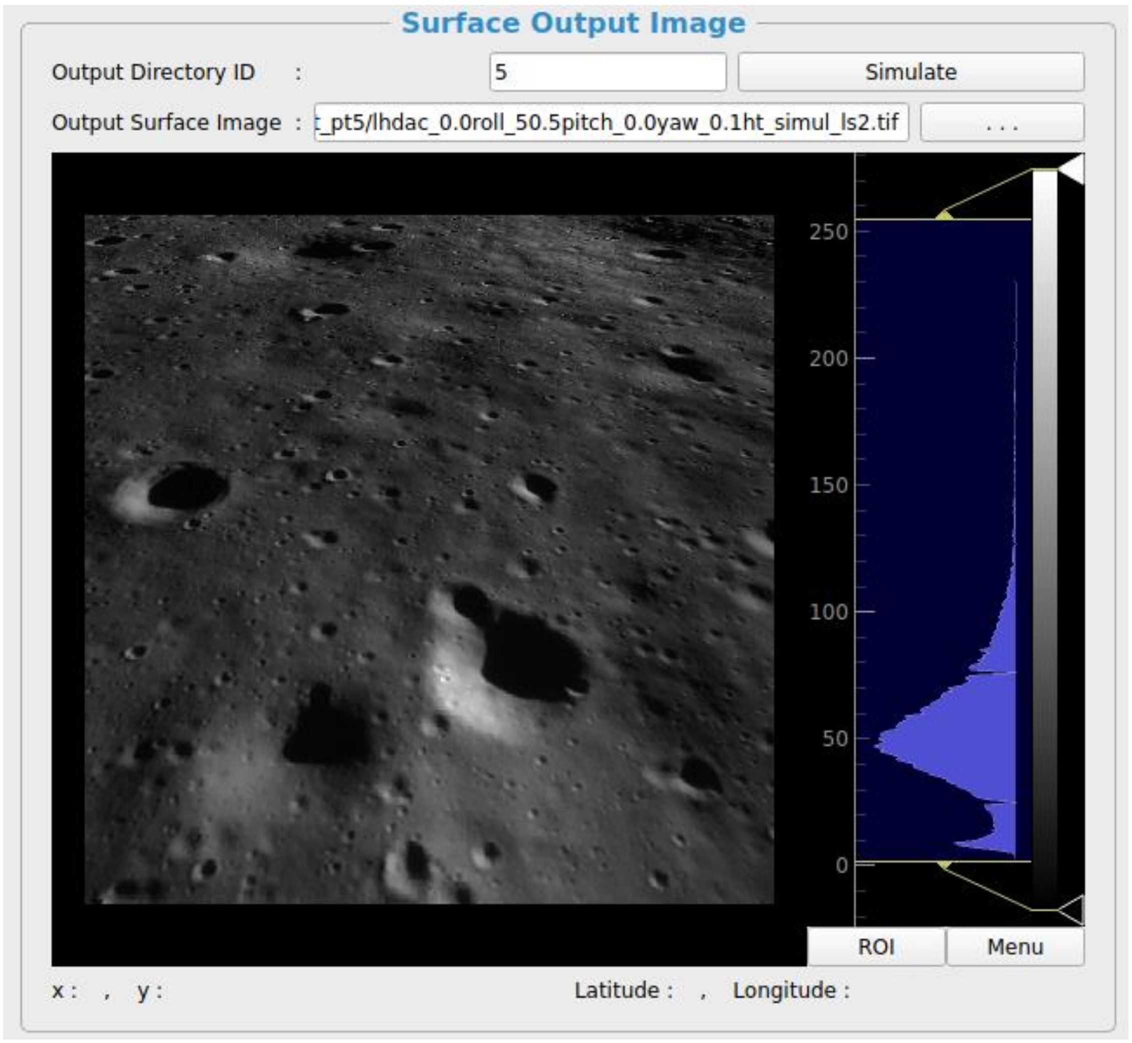
Upper Left Corner : (-69.405353,32.346763)   Upper Right Corner : (-69.418641,32.362467)
Lower Left Corner : (-69.405353,32.387887)   Lower Right Corner : (-69.418641,32.372168)

Surface Output Image

Output Directory ID : 5   Simulate
Output Surface Image : t_pt5/lhdac_0.0roll_50.5pitch_0.0yaw_0.1ht_simul_ls2.tif

x : , y :   Latitude : , Longitude :

Figure 6.17 Example_3


Surface Output Image

Output Directory ID : 5   Simulate
Output Surface Image : t_pt5/lhdac_0.0roll_50.5pitch_0.0yaw_0.1ht_simul_ls2.tif

x : , y :   Latitude : , Longitude :

Figure 6.18 Example_3 Output

7 TESTING

7.1 TESTING PLAN /STRATEGY

The testing phase ensures that the image simulation system functions correctly and meets the intended objectives. The strategy involves:

7.1.1 Testing Types Used

- Unit Testing – Individual components (e.g., image loading, parameter inputs, footprint preview) are tested separately.
- Integration Testing – Ensures that different modules (e.g., GUI, parameter selection, simulation engine) work together correctly.
- Functional Testing – Verifies that all features (e.g., single/multi-point simulation, marker placement) perform as expected.
- Performance Testing – Checks system efficiency with different image sizes and parameter variations.
- User Acceptance Testing (UAT) – Ensures the system meets user requirements for camera testing and design.

7.1.2 Testing Environment

- Platform: Linux
- Software Dependencies: Python, PyQt5, PyQtGraph, GDAL, NumPy
- Hardware: System with at least 8GB RAM and multi-core processor

7.2 TEST RESULTS AND ANALYSIS

Multiple test cases were executed to verify the system’s performance and accuracy. The following table summarizes key test cases:

7.2.1 Test Cases

Test ID	Test Condition	Expected Output	Actual Output	Remarks
TC-01	Load GeoTIFF image	Image loads successfully	Image loads correctly	Pass
TC-02	Enter valid camera parameters	Parameters accepted and applied	Parameters stored and used	Pass
TC-03	Enter invalid camera parameters	Error message displayed	Error handled correctly	Pass
TC-04	Single-point marker placement	Marker appears at correct lat-long	Marker placed accurately	Pass
TC-05	Multi-point marker placement	Two markers placed correctly	Markers appear as expected	Pass
TC-06	Footprint preview	Correct footprint displayed	Footprint aligns with expected region	Pass
TC-07	Run image simulation	Simulated image is generated	Image output is accurate	Pass
TC-08	Save simulated image	Image saved in directory	File saved successfully	Pass
TC-09	Clear parameters	All fields reset	Fields cleared properly	Pass
TC-10	Performance with high-res image	Processing within reasonable time	Acceptable performance	Pass (minor delay)

Table 7.1: Test Cases for Image Simulation System

7.3 ANALYSIS OF TEST RESULTS

- Functionality: All core features performed as expected, with accurate image simulation.
- Usability: The GUI is intuitive, with smooth user interactions.
- Performance: The system handles various image resolutions well, but higher-resolution images slightly increase processing time.
- Error Handling: Proper validation prevents invalid inputs, improving system robustness.

8 CONCLUSION AND SUMMARY

8.1 CONCLUSION AND SUMMARY

The image simulation system was developed to evaluate how a camera would capture images in an unknown environment. By incorporating camera-specific parameters such as detector height, width, field of view (FOV), focal length, and altitude, the system provides an effective way to analyze and design cameras before actual deployment.

The system meets its objectives by providing a useful tool for Image simulation and being helpful for studying images.

8.2 LIMITATIONS

- Processing Time for High-Resolution Images: When working with very large GeoTIFF images, processing can take longer than expected. Optimizations in computation can help reduce delays.
- Limited Camera Parameter Customization: The system currently supports predefined parameters and manual input. Future versions could allow users to import real-world camera calibration data.
- No Real-Time 3D Visualization: The system operates in a 2D simulation environment. Adding a 3D rendering option could improve visualization and understanding of camera perspectives.
- Georeferencing Accuracy: While the system accurately places markers based on latitude-longitude coordinates, minor discrepancies may arise due to projection transformations.
- No External Data Integration: Currently, the system processes only user-provided images. Future enhancements could support integration with external satellite data sources.