Integrating Advanced Visualization and Automated HPC Workflows for GNSS Prediction in Urban Air Mobility

Elena Deckert - NASA Langley Research Center Intern
Julian Gutierrez, Russell Gilabert, Evan Dill – NASA Langley Research Center

21 July 2025

AIAA Aviation, Las Vegas

Outline

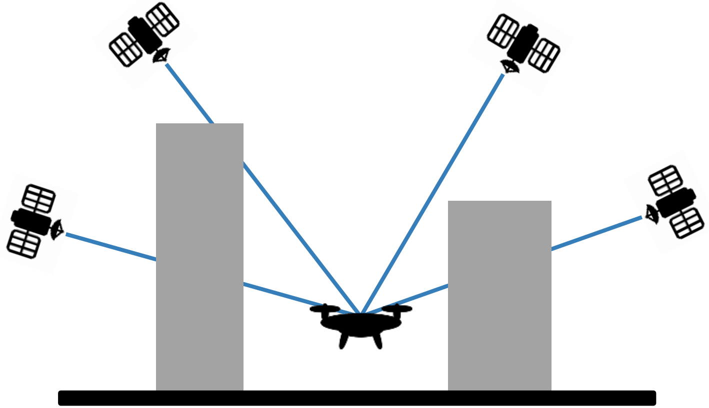
- Problem Statement
- System Workflow
- Performance Analysis
- Conclusions

Problem Statement

PASSENGER EMERGENCY
Passenger Health Monitoring & Airspace Prioritization
FLIGHT OVER PEOPLE/MOVING VEHICLES
3rd Party Risk Modeling
TERRAIN COLLISION AVOIDANCE
Proximity to Threat
GPS DEGRADATION
GPS Degradation Models/APNT Solutions
RF INTERFERENCE
RF Interference Models
Noise Abatement Zone
Original Route
TRAFFIC COLLISION AVOIDANCE
DAA Safety Monitor
Non-cooperative
Modified Route
Pre-flight Safety
OBSTACLE AVOIDANCE
Enhanced Obstacle Database/Detection
Hospital Vertiport
+10 min
Limited vehicle performance
4DT ROUTE CONFLICT
Airspace Density Safety Service
WEATHER
Advanced Weather Models
Post-flight Safety
Continue Operation
Emergency Landing
VEHICLE SYSTEM FAILURE
Vehicle Health Monitors
In-flight Safety
IASMS Capability Coverage
--Monitor, Assess, Mitigate
--Pre-flight, In-flight, Post-flight/Off-line
--Reactive, Proactive, Predictive

Problem Statement

- The quality of a GNSS based position solution is largely correlated to:
  - The number of satellites in Line of Sight
  - The geometry of the available satellites: dilution of precision (DOP)
  - The correctness of the available signals
- In low altitude flights, it is common to have objects (e.g., buildings, terrain, foliage) obstruct the direct line of sight

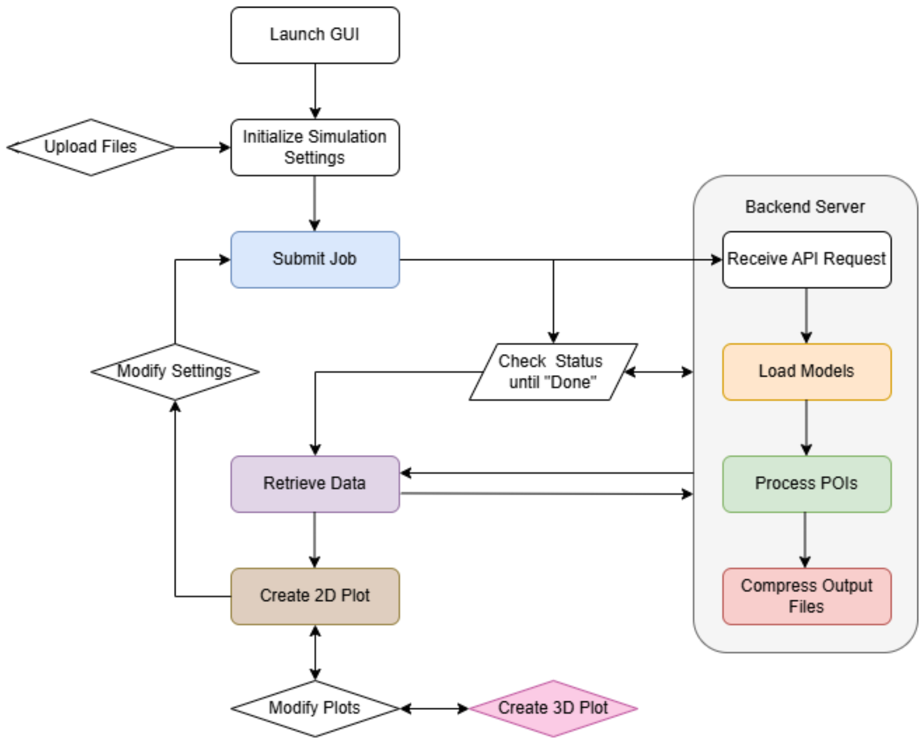


We introduce an automated HPC workflow backend and a GUI producing advanced visualizations to help analyze the risk of GNSS performance degradation.

System Workflow

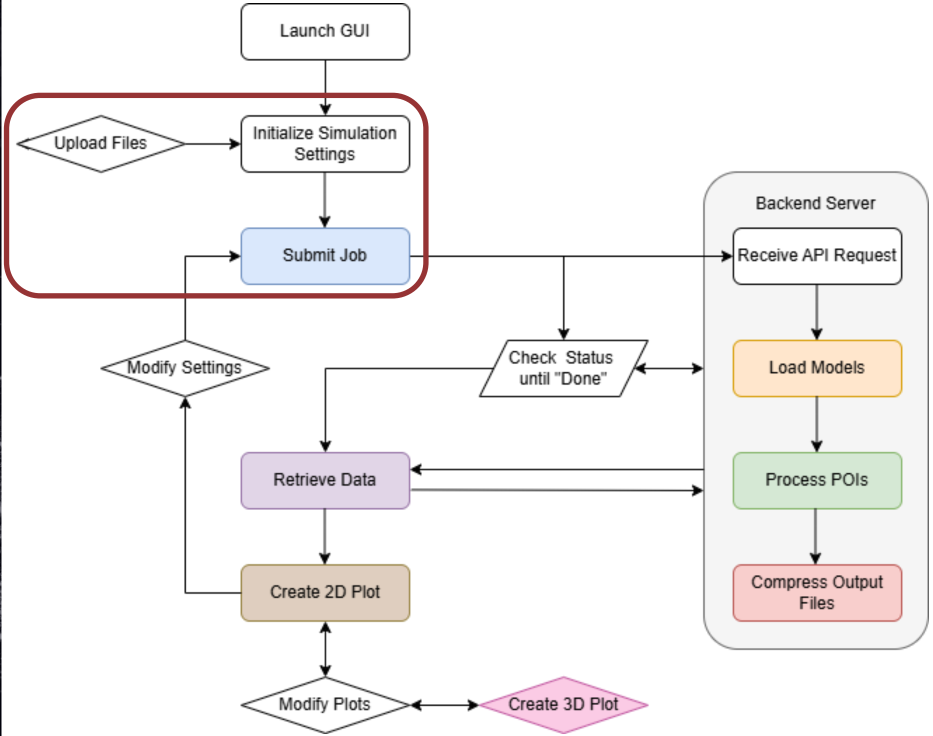
A GNSS Performance Analysis Tool

- GNSS safety analysis of flight trajectories and operational zones for risk mitigation
- Leverages HPC to predict line-of-sight (LOS) GNSS availability
- Visualizes multidimensional data to gain an understanding of hazardous areas
- Implemented as a wrapper over the NASA NavQ tool



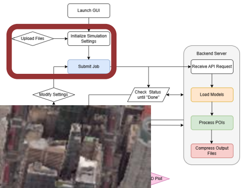
Simulation Controls

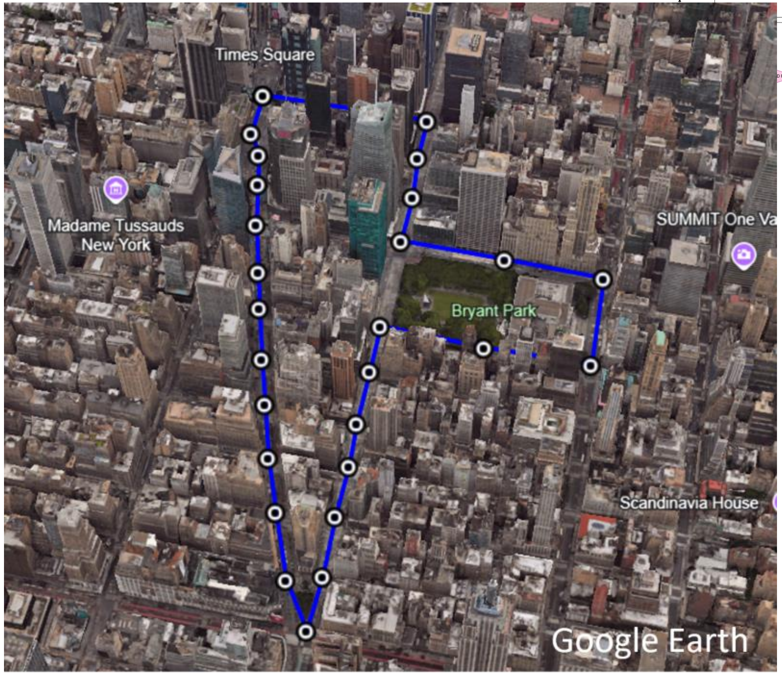
- The mission planner can initialize the simulation settings
- Specify the Points of Interest (POI) by adding them manually or uploading a POI list
    - Fixed point analysis
    - Interpolated flight path



Simulation Controls

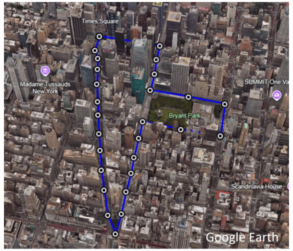
- The mission planner can initialize the simulation settings
- Specify the Points of Interest (POI) by adding them manually or uploading a POI list
  - Fixed point analysis
  - Interpolated flight path

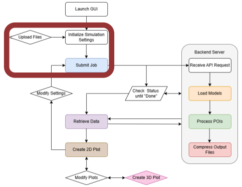


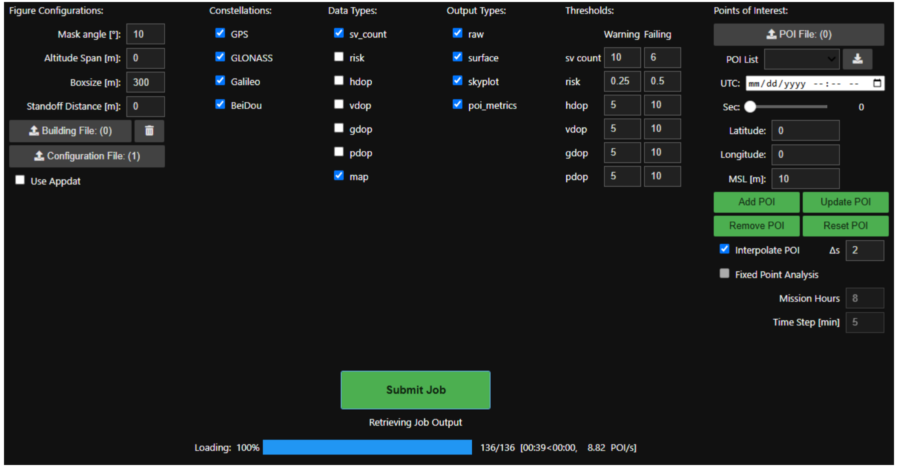


Simulation Controls

- Configuration options:
  - Satellite constellations
  - Data types (SV Count, DOP, Risk, Map)
  - Output types for visualization
  - Threshold values (warning/failure levels)

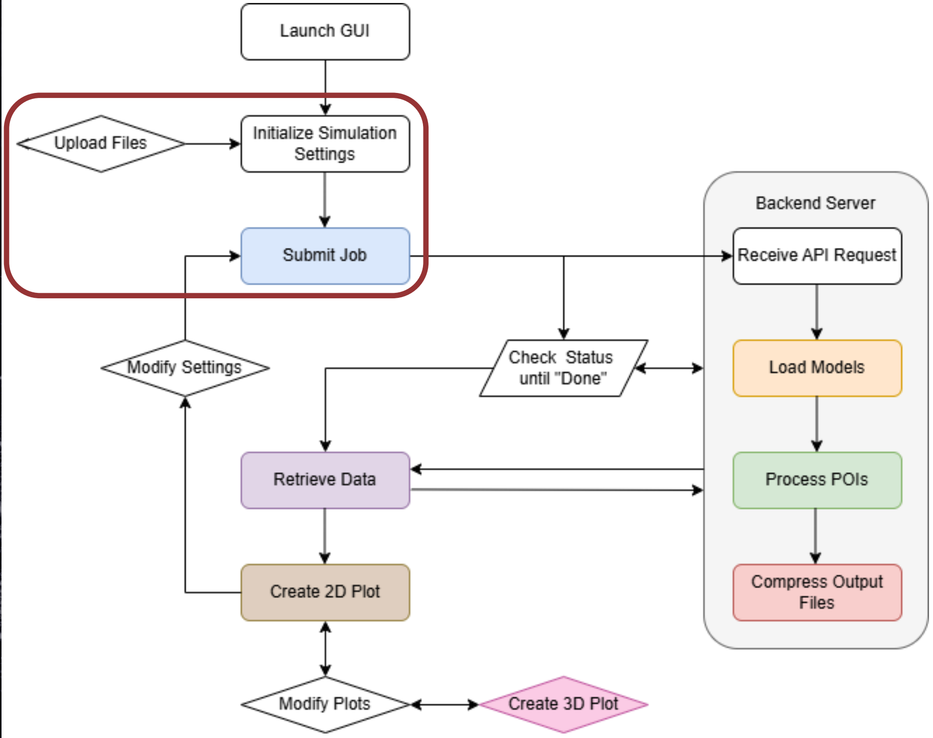






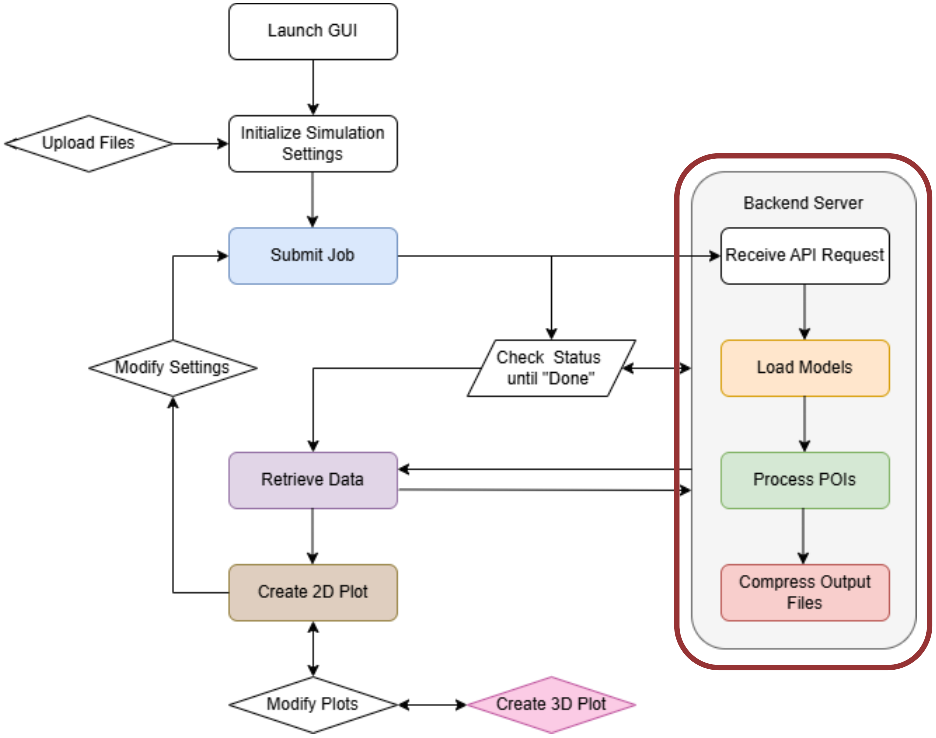
Simulation Controls

- Configuration options:
  - Satellite constellations
  - Data types (SV Count, DOP, Risk, Map)
  - Output types for visualization
  - Threshold values (warning/failure levels)



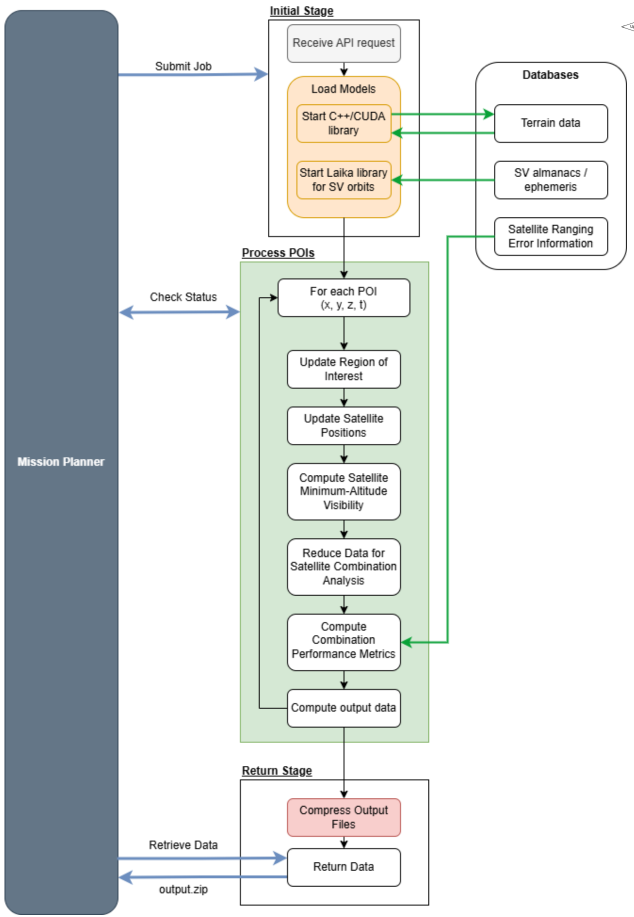
Backend Processing

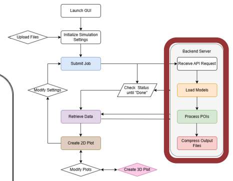
- Backend as a wrapper on NavQ
- Deployed via Kubernetes, Docker containers, with a Flask API
- Backend Capabilities:
  - 3D map manipulation capabilities
  - Laika instantiation for satellite information
  - Payload verification
  - Error handling strategies
  - NavQ instantiation process



Backend Processing

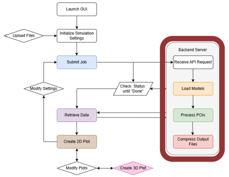
- Creates a discretized time-varying 3D volume that computes the estimated quality of GNSS-based position solutions
- A GPU-accelerated function computes the minimum height for a satellite to be LOS
- A parallelized CPU function processes the unique satellite combinations to determine their respective performance metrics

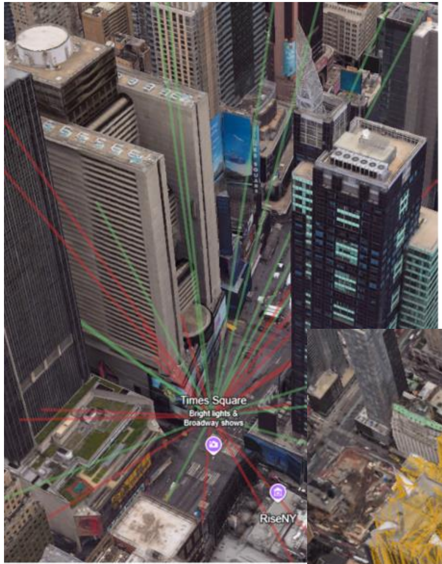


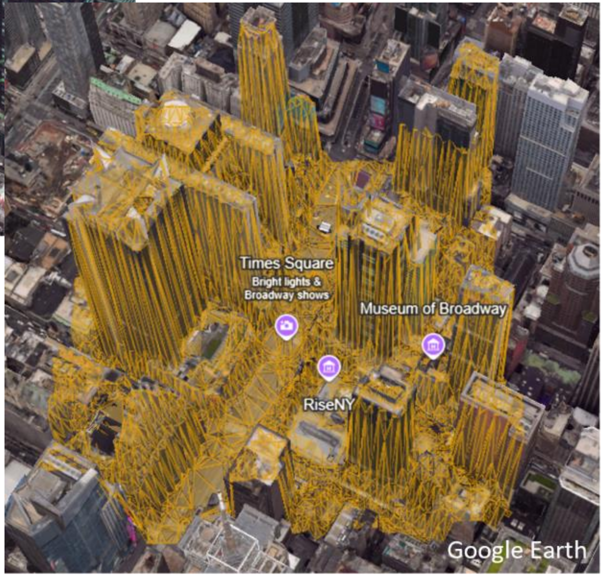


Backend Processing

- Creates a discretized time-varying 3D volume that computes the estimated quality of GNSS-based position solutions
- A GPU-accelerated function computes the minimum height for a satellite to be LOS
- A parallelized CPU function processes the unique satellite combinations to determine their respective performance metrics

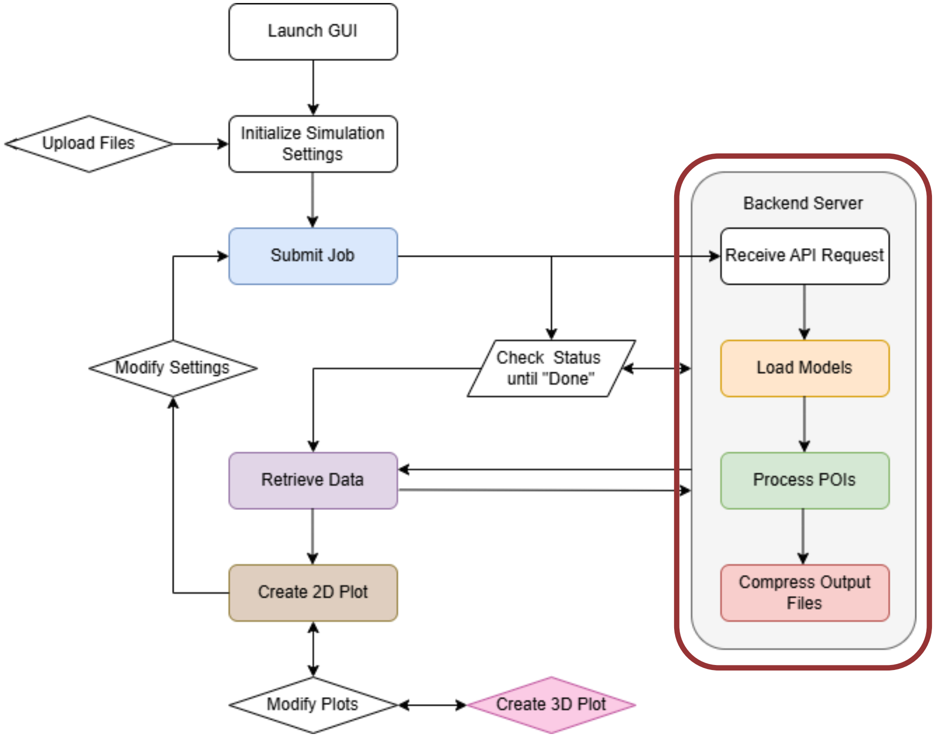






Backend Processing

- Creates a discretized time-varying 3D volume that computes the estimated quality of GNSS-based position solutions
- A GPU-accelerated function computes the minimum height for a satellite to be LOS
- A parallelized CPU function processes the unique satellite combinations to determine their respective performance metrics



Final Visualizations

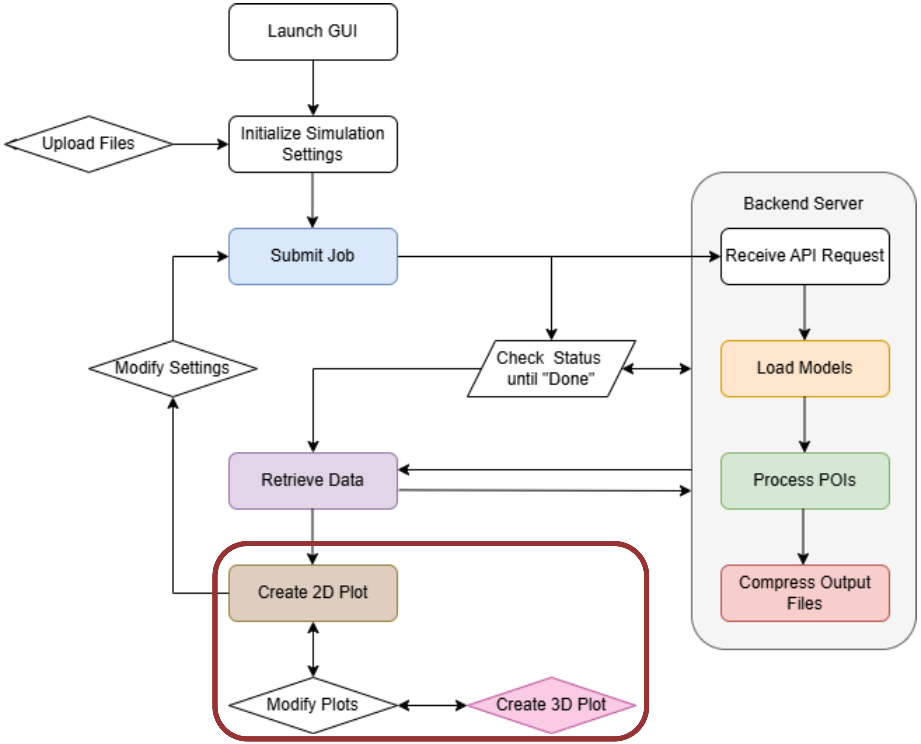
- Create plots to illustrate the satellite coverage for specific areas or paths highlighting potential blind spots

Job Summary

Start time: 2024-12-02 12:42:23
Total time: 133.5408 s
Model load time: 25.0274 s
Average POI processing time: 0.301 s
File compression time: 79.3169 s
Client IP: 172.18.0.1
Output file size: 279.9093 MB

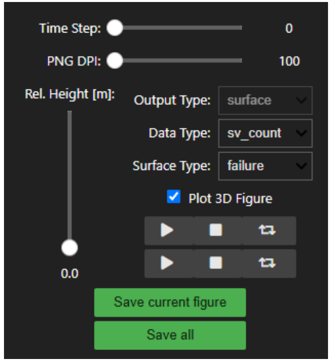
Percentage of POIs not meeting the **sv_count** threshold:

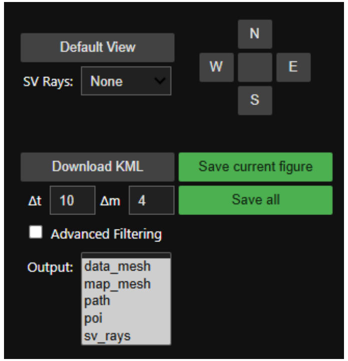
Failure: 5.15 %
Warning: 38.14 %

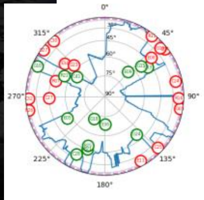


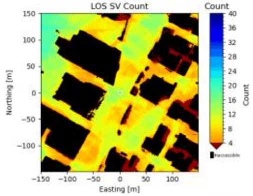
Final Visualizations

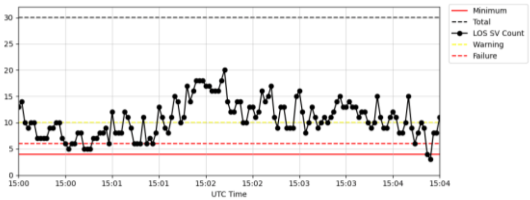
- Create plots to illustrate the satellite coverage for specific areas or paths highlighting potential blind spots

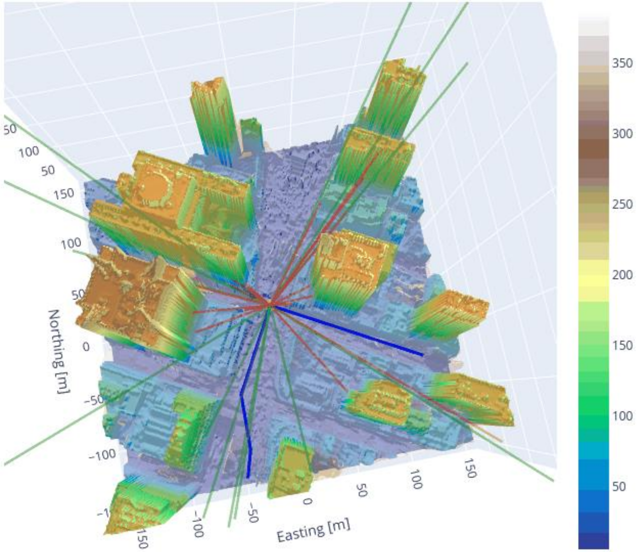






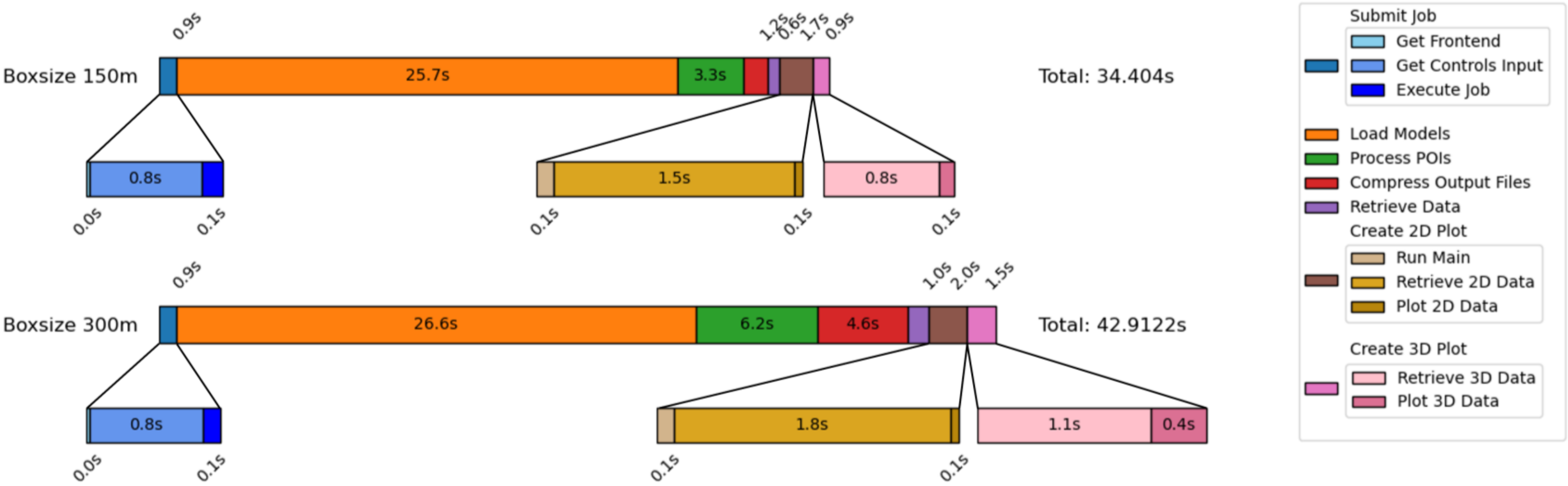




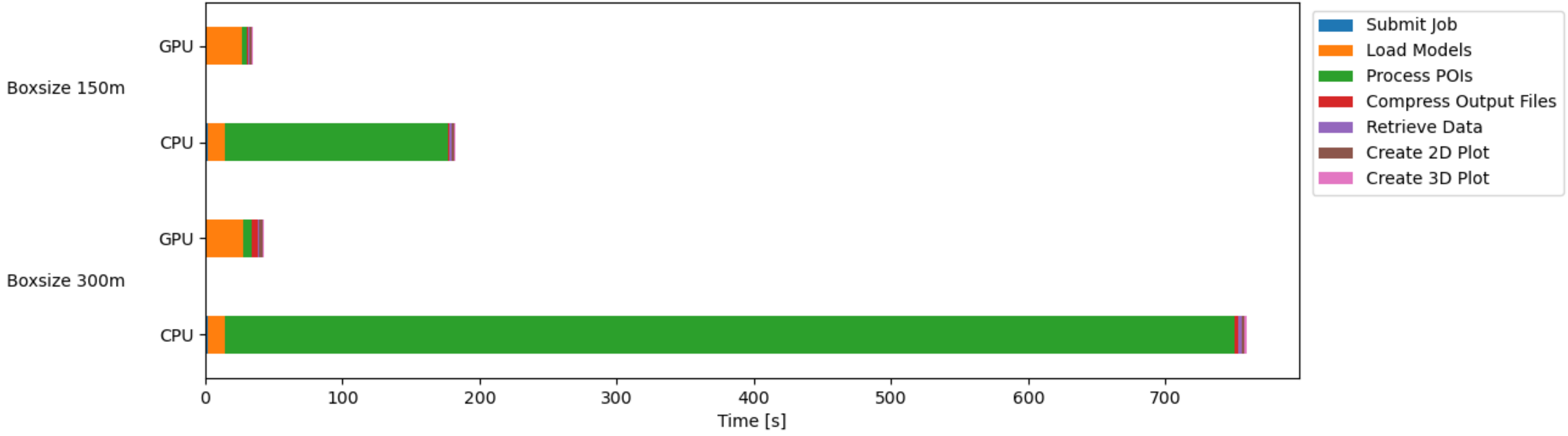


Performance Analysis

Computational Breakdown of Processing steps



Comparing the GPU and CPU Implementation



The **GPU implementation** is necessary to enable the analysis of flight paths for real missions in reasonable time.

Conclusions

Conclusions

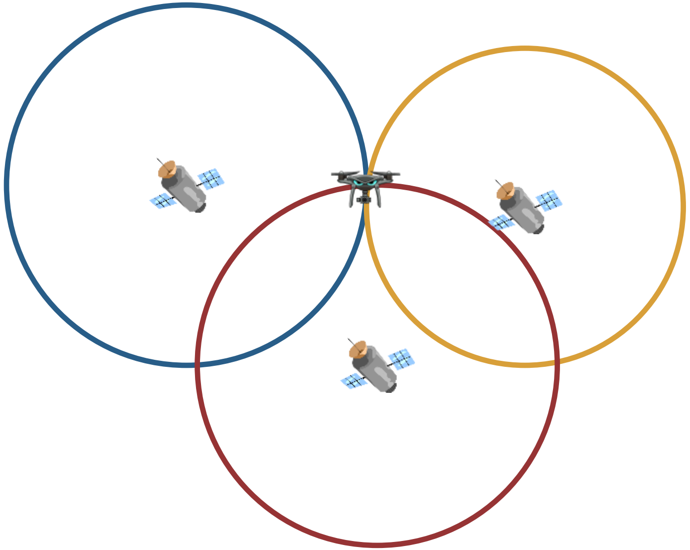
- Introduce a backend and frontend integrating an automated HPC workflow to produce advanced visualizations for GNSS risk mitigation analysis in UAM
- Demonstrate the importance of GPU-accelerated computations in supporting real-time GNSS prediction and flight path analysis for UAM applications
- Showcase the potential benefits of HPC adoption and serves as a steppingstone for future investigations into optimizing GPU utilization
- Future work:
  - Optimizing the Laika library to reduce loading times
  - Implementing real-time communication capabilities between the frontend and backend

Thanks!

Any questions?

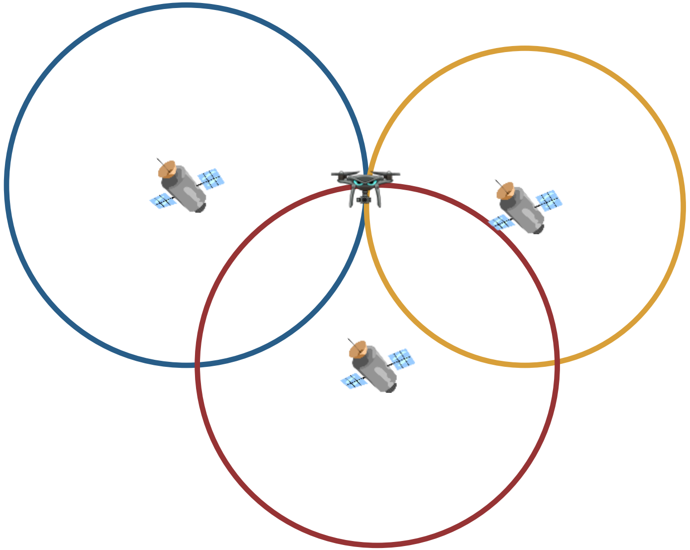
GNSS Basics

- Signals are Time-tagged and sent from satellites
- The distance between the satellite and the receiver (pseudo-range) is estimated using the Time-of-Flight (signal propagation time)
- We know the satellite's location based on Ephemeris data and the receiver will be located somewhere on a sphere around it
- The point of intersection of the spheres is the receiver's estimated location



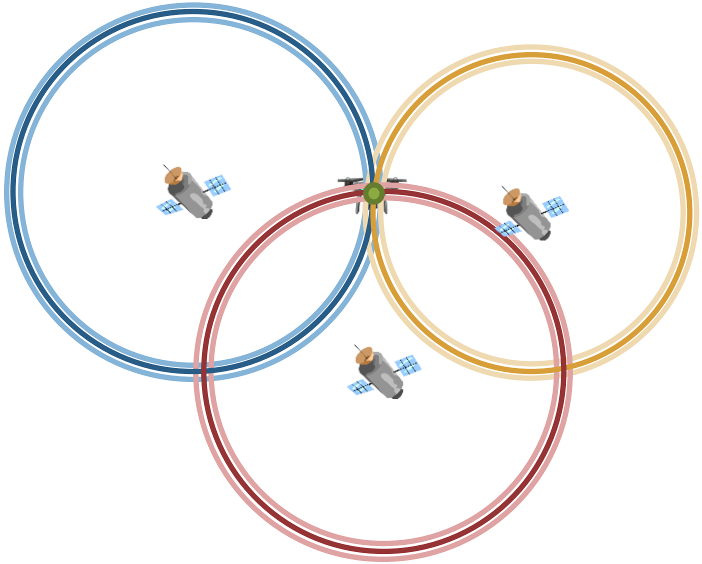
GNSS Basics

- GNSS requires incredibly precise timing to accurately measure the time of flight
- GNSS measurements are not true ranges, they are **pseudo-ranges** (true range + Rx clock errors)
- Intuitively, we would need 3 satellite measurements (equations) to solve for a 3D position (3 variables)
- If we have 4 measurements, we can address the Rx clock error



GNSS Basics

- We can expect a margin of error in the measurements from each satellite due to a multitude of factors:
  - E.g., atmospheric errors, orbit errors, relativistic errors, etc.
- In this example, we can see a clear intersection point
  - This is due to a good geometry of the satellites



GNSS Basics

- If we have poor satellite geometry, our position estimation error increases (given by the size of the green ellipse)
- If a new satellite becomes available, we can improve our estimation error if it improves the geometry
- In other words, the more satellites we can use, the better the quality of our estimates