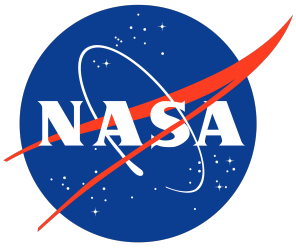
NASA/TM–20240009932



A High-Performance Computing Predictive GNSS Performance Monitor for Autonomous Air Vehicles in Urban Environments

Julian Gutierrez, Steven Young, Andrew Moore, Russell Gilabert, Evan Dill
Langley Research Center, Hampton, Virginia

Emily Bates, Matthew Peretic, Ken Schmitt, Arthur Scholz
The MITRE Corporation, Massachusetts

August 2024

NASA STI Program Report Series

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NTRS Registered and its public interface, the NASA Technical Report Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

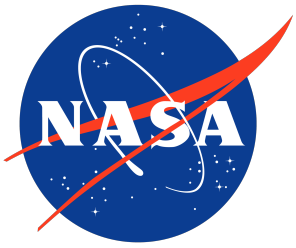
- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.
- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.
- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI Program, see the following:

- Access the NASA STI program home page at `http://www.sti.nasa.gov`
- Help desk contact information: `https://www.sti.nasa.gov/sti-contact-form/` and select the "General" help request type.

NASA/TM–20240009932



A High-Performance Computing Predictive GNSS Performance Monitor for Autonomous Air Vehicles in Urban Environments

Julian Gutierrez, Steven Young, Andrew Moore, Russell Gilabert, Evan Dill
Langley Research Center, Hampton, Virginia

Emily Bates, Matthew Peretic, Ken Schmitt, Arthur Scholz
The MITRE Corporation, Massachusetts

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

August 2024

Acknowledgments

The authors thank Johnson Carroll from the MITRE corporation for his insights and recommendations during the revision of this manuscript. Additionally, the authors thank Erik Lundberg and Keith McDonald from the MITRE Corporation and Tanner Slagel and Lauren White from NASA Langley Research Center for their technical review.

This technical data was produced in part for the U. S. Government under Purchase Order 80NSSC23PA865, and is subject to the Rights in Data-General Clause 52.227-14, Alt. IV (DEC 2007). **Approved for Public Release; Distribution Unlimited. Public Release Case Number 24-0475.**

Available from:

NASA STI Program / Mail Stop 150
NASA Langley Research Center
Hampton, VA 23681-2199

Abstract

This report offers analysis and design insights for leveraging High-Performance Computing (HPC) to predict line-of-sight (LOS) Global Navigation Satellite System (GNSS) availability in a city. This work is motivated by the emerging fields of Advanced and Urban Air Mobility (AAM/UAM), where regulatory authorities are seeking city-scale, meter-resolution risk forecasting in order to safely integrate new flight missions with existing urban life and infrastructure. This work addresses the technical challenge of efficiently computing urban GNSS satellite visibility to predict GNSS performance metrics under these requirements.

We present a new HPC-optimized shadow casting algorithm variant as a ray-based approach to forecasting satellite visibility. We apply this algorithm variant in a software-defined prognostic service which generates a GNSS navigation risk-correlated map as a path planning-style potential field. We detail dominant computational burdens, viable simplifying assumptions, and different algorithmic implementations, intending to demonstrate a baseline of computation time needed by each stage in such a service. We conclude by analyzing the prototype service's prediction accuracy compared to receiver data from Corpus Christi, Texas. This informs design trade-offs along the dimensions of hardware, computation time, and tolerable forecasting error (including proportions of false positives and false negatives).

Contents

List of Tables

List of Figures

1 Introduction

The emergence of Advanced Air Mobility (AAM) and Urban Air Mobility (UAM) technologies presents regulatory authorities and the aviation community with the challenge of safely integrating new operational paradigms. It is well-known that urban environments induce complex and hard-to-model behaviors in Global Navigation Satellite System (GNSS) receivers due to non-line of sight (NLOS) and multipath effects (Appleget and Bartone, 2019; Betz, 2015). Although the body of knowledge for real-time receiver-side NLOS/multipath compensation continues to grow, urban GNSS-assured receivers may not always be possible, practical, or commercially available, especially in the low Urban Air Mobility Maturity Levels (UMLs) (Goodrich and Theodore, 2021; Patterson et al., 2021). Thus, as demonstrated in fixed-wing civil aviation, regulatory authorities and mission planners can turn to risk prediction techniques to construct operations that, backed by quantitative analysis, minimize the harmful NLOS/multipath effects ingested by the AAM/UAM navigation system, and potentially bound the possible error in the AAM/UAM navigation system over the operation.

A survey of related work (Section 2) indicates that forecasting navigation risk-correlated information over an urban-scale region at AAM/UAM-relevant resolutions is possible, but it is computationally challenging. The problem has a) a minimum of four dimensions in path planning applications, b) a need for near-real time forecasting (for instance, to account for unexpected changes in satellite health or in urban terrain), and c) computationally time-consuming satellite visibility calculations.

While the research community has succeeded in studying urban navigation risk despite these computational challenges, the computational burden may be limiting technical exploration. Some papers, such as (Groves et al., 2015), note outright that computing satellite visibility is intensive compared to other steps in urban navigation. Other studies reduce the computational workload by reducing the number of intersection tests between satellite transmission ray and 3D model, or by assessing navigation risk over a trajectory rather than a hyper-volume. In both research prototypes and commercial solutions, descriptions of the exact satellite visibility modeling technique used are often incomplete and, therefore, not available for replication, verification, and adoption by the research and AAM/UAM communities. Thus, there is a need for replicable satellite visibility forecasting algorithms that forecast navigation risk in four dimensions on computing resources of a practical cost.

Fortunately, as we show, high performance computing (HPC) techniques and equipment have an affinity for the AAM/UAM navigation risk forecasting problem. Leveraging HPC is a matter of refactoring parts of the navigation risk processing chain (for example, changing the order of execution of certain steps, selecting specific memory constructs, and inverting parts of the problem), and of choosing trade-offs that optimize for both the GNSS-domain and the HPC-domain objectives.

Our primary contribution in this paper is a new variation of shadow casting. Shadow casting inverts a large portion of the ray tracing algorithm, which can offer computational benefits in certain types of problems. Our shadow casting variant is a new design which, unlike raycasting implementations in other fields, is optimized both for HPC and GNSS concepts. We believe that this variation on the shadow

casting algorithm is unique in the GNSS literature. Our companion conference papers (Dill et al., 2021; Moore et al., 2023a) presented results using this algorithmic variation, but this paper provides the first detailed exposition of the algorithm.

We then apply the shadow casting algorithm by prototyping an end-to-end AAM/UAM navigation risk-correlated map forecasting tool. For brevity, this navigation quality tool is referred to herein as NavQ. While there are brief reports of similar capabilities in the literature, as Section 2 will discuss, these reports detail neither the compute time nor the computing resources required to produce their results. As a result, there lacks a general assessment on the order of magnitude of computation time for common stages in the simulation needed to enable AAM/UAM navigation risk map forecasting. Thus, as a second contribution, we profile the performance of the NavQ prototype, analyzing the computation time for different stages of the process and demonstrating the overall feasibility of this approach. For this contribution to be as broadly applicable as possible, we impose the following objectives, requirements, and design constraints on NavQ:

Requirements:

- Output: a discrete, 4D map (two horizontal dimensions, the vertical dimension, and a time dimension) of a risk-correlated forecasting metric for each value in the map.
- Spatial scale: the position components of the results have one-meter resolution, chosen as roughly half the scale of the smallest rotor radius of the UAM concept vehicles described in (Johnson and Silva, 2022).
- Time scale: risk-correlated map generation for flight path planning-relevant regions of an American city-scale region computed within three seconds, following guidelines proposed in (Moore et al., 2023a).
- Confidence: 95% accuracy of NLOS/LOS prediction differing by two SVs or less, following guidelines proposed in (Moore et al., 2023a).

Constraints:

- GNSS Inputs: a valid almanac as the only GNSS-related information available. No real-time pseudoranges or residuals; such would require tight integration with the AAM/UAM receiver, and some commercial products may not expose these observables, limiting the applicability of this study.
- Terrain inputs: a 3D city model as the only city-related information available. No real-time camera or LiDAR; such would require tight integration with the AAM/UAM receiver, and not all commercial receivers include a camera or LiDAR, limiting the applicability of this study.

We hope that this study will allow this work to serve as a baseline for forecasting navigation risk for AAM/UAM path planning capabilities. For this reason, we also choose to avoid the use of machine learning (ML). The high dimensionality of ML

models can obscure details in the model's processing, making it difficult to profile the processing at a high resolution. However, supplementing or replacing stages of the NavQ prototype with ML models may be interesting future extensions to compare with the current prototype.

Using NavQ as a baseline, we discuss trades in the space of GNSS modeling assumptions, HPC hardware requirements, and computation time performance throughout Section 3. Section 4 presents computation time and satellite visibility prediction accuracy analysis from NavQ, including comparisons between processing techniques and processing demands.

2 Background

There is a rapidly growing body of knowledge on the topic of GNSS receiver performance in urban environments. Many studies require modeling regions of the city which are in shadow from the perspective of the GNSS satellites. When computing shadow areas, of the works that describe their algorithm in detail, we find that the shadow-computing algorithms either use ray tracing methods which indirectly provide shadow information only after determining whether the signal is LOS, or projecting polygons onto the horizontal plane and testing vertices for boundary points. We have not found prior discussion in the GNSS literature of the shadow casting algorithm that we propose. Additionally, we find a marginal amount of study explicitly or implicitly imposing a similar combination of requirements as detailed in Section 1. The following subsection surveys the relevant body of literature. Section 2.2 recaps the concept of dilution of precision (DOP) and surveys its well-established use as a risk-correlated metric.

2.1 Related Work

Perhaps the closest work on urban satnav shadow computing is (Bradbury, 2007). This simulator ingests a facet-based 3D model. Each polygon in the map is projected onto the horizontal plane, then an unexplained algorithm identifies the outer vertices to produce a single polygonal region. However, this approach does not consider the vertical dimension, and a naïve extension of stacking up horizontal layers at varying altitudes would recompute a large amount of information from prior layers. In contrast, our shadow casting approach implicitly determines satellite visibility at all altitudes across the whole map. Bradbury's work is also only designed for evaluating a single horizontal position-time state per run, and does not discuss computation time.

Also of note, shadow casting is mentioned under the name raycasting in (Groves et al., 2015). The paper briefly states that shadow casting can improve NLOS/LOS decision computation time with a trade-off of increased storage and certain quantization errors, but does not investigate the topic further. Some works on the topic of the shadow *matching* position algorithm do describe their algorithm for modeling urban satnav shadows (Ben-Moshe et al., 2011; Bhamidipati et al., 2021; Bradbury, 2007; Neamati et al., 2023; Zhong and Groves, 2023), but these do not propose a shadow *casting* approach to the satellite visibility determination. The aforementioned papers

instead perform vertex boundary tests, use receiver measurements, perform line-of-sight ray tracing (computing the shadows indirectly), or pre-generate a custom map representation.

An increasing number of papers forecast satellite visibility along automotive trajectories or generate maps exclusively in the horizontal plane, including (Costa, 2011; Jakobsen and Jensen, 2013; Kleijer et al., 2009; Lee et al., 2008; Nagai et al., 2020a; Suh et al., 2004). Recently, some papers have added a vertical dimension for AAM/UAM trajectories (Bijjahalli et al., 2018; Causa and Fasano, 2021; Pongsakornsathien et al., 2020). These are not modeling a full hyper-volume. None of these papers discuss computation time. These papers either do not mention the satellite visibility model used, or they use a black-box commercially available RF propagation simulator.

Among the papers that do mention computational optimizations, these take a very different form. Papers including (Bradbury, 2007; Ziedan, 2017) propose methods for reducing the number of intersection tests to run. (O'Connor et al., 2021) does mention GPU acceleration, but it is left as a black box with only overall runtime statistics. It does not detail any specific algorithm for visibility forecasting. (Tadic et al., 2013) shows that GPU ray tracing outperforms CPU ray tracing when modeling several multipath bounces, but the CPU and GPU computation time is nearly identical in a zero-bounce (LOS visibility) scenario, and the ray tracing algorithm and its means of GPU acceleration is not detailed.

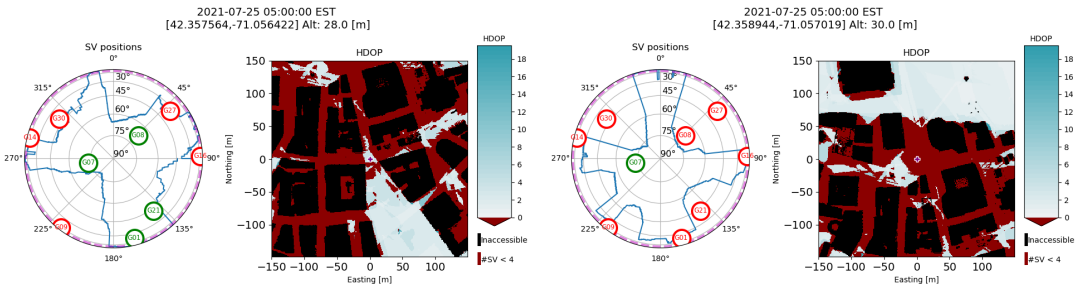
Other works still have fundamentally different design requirements. Many determine satellite visibility from receiver observables or other sensor modalities, such as (Ali et al., 2012; Furukawa et al., 2020; Sanromà Sánchez et al., 2016; Tokura and Kubo, 2016). Others use observables to train ML models, such as (Smolyakov et al., 2020; van Diggelen, 2021), which estimates satellite visibility (among other effects such as received carrier-to-noise and pseudorange corrections) from receiver measurements and features from the 3D city model. Others still perform similar visibility modeling to instead produce pseudorange error corrections or bounding (Bourdeau et al., 2013; Drevelle and Bonnifait, 2011; Kbayer and Sahmoudi, 2018; Obst, 2014).

While studies to date have advanced understanding of low-altitude navigation risk forecasting, key deficiencies limit their applicability. First, studies to date have commonly simulated trajectories rather than hyper-volumes. AAM/UAM path planning utilizes 4D maps to generate mission trajectories. Second, the naïve or generalized ray tracing commonly employed is inefficient, as it often repeats computational operations, and it does not directly compute the primary information that the urban satnav visibility problem requires. The shadow casting method detailed in Section 3 offers one specific algorithm to address this. Lastly, the lack of discussion on total computation time may present the researcher with unexpected roadblocks, should some stage of the processing take prohibitively long. Section 4 presents some baseline information regarding practical computation time demands of AAM/UAM navigation risk forecasting.

2.2 DOP as a Navigation Risk-Correlated Metric

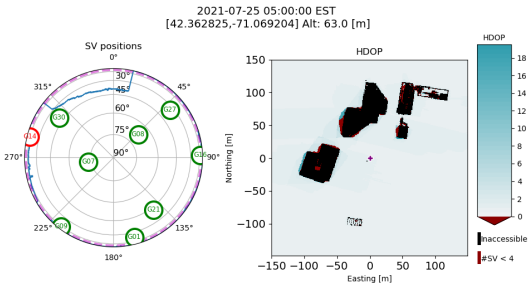
As a translation from satellite visibility prediction to a navigation risk-correlated map intended for AAM/UAM path planning, NavQ employs DOP. Given the requirements and constraints detailed in Section 1, DOP is defensibly the most well-established risk-correlated metric. DOP has long been employed to study GNSS receiver navigation performance in urban regions including Seoul, South Korea (Lee et al., 2008), Hong Kong (Ji et al., 2010), Indianapolis, USA (Anyaegbu and Hansen, 2022), and Chicago, USA (Nagai et al., 2020b). A lower DOP value is considered a better solution, and values under ten are often considered geometries that may provide sufficient positional accuracy for conventional navigation applications (Betz, 2015).

While the Global Positioning System (GPS) alone specifies that the position component of dilution of precision (PDOP) will be, on average, a value of six or smaller for 98% of a sidereal day (Dunn, 2022), dense urban environments can experience significantly worse DOP, including large stretches of unavailability. As an example, Figure 1 samples the GPS-only horizontal DOP (HDOP) at various times and location in urban regions of Boston, Massachusetts, USA. Often times, a low altitude, single constellation receiver will be unable to solve for its own position. Even for a multi-constellation receiver, should there be four or more visible satellites, the sky occlusions impose a very poor geometry, and the resultant errors could be unacceptably poor.

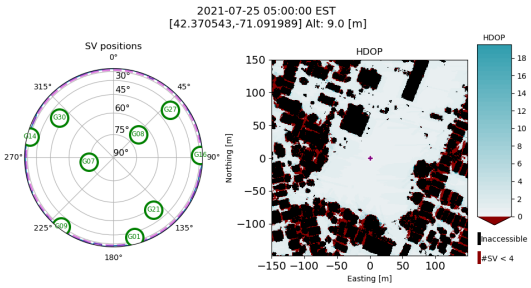


(a) Location 1: Financial District

(b) Location 2: Financial District



(c) Location 3: Top of Hospital Building



(d) Location 4: Baseball Field

Figure 1: Skyplot and HDOP data at locations within Boston, USA. Altitude is given from mean sea level (MSL).

DOP is not a standalone assessment of navigation risk. However, meaningfully

predicting GNSS receiver errors expressed in the position domain is an area of study unto itself. For the scope of this work, given the technical maturity of DOP and its history of application to the field of urban navigation, we find DOP to be sufficient for use as a risk-correlated metric for NavQ's stated objective. Extending NavQ to incorporate a technique which directly forecasts navigation risk is suitable incremental future work, and we note how such an extension might integrate where appropriate in Section 3.

3 Computational Optimizations

This section introduces the design of NavQ. Section 3.1 defines the terms, variables, and processing flow for NavQ. Section 3.2 introduces the required input data objects and pre-processing. Section 3.3 proposes a formulation of the satellite visibility problem to improve computational efficiency, and details both a naive line-of-sight algorithm and the shadow casting algorithm variation under this formulation. Finally, Section 3.4 shows how to generate the navigation-risk correlated map as parallelized lookup operations, and discusses the ensuing tradeoffs.

3.1 Definitions

Figure 2 presents a block diagram of the NavQ prototype's computational processing. The following subsections discuss in turn the computational processing of the input stage, the ray-based satellite visibility calculations, and the output stage.

For the remainder of the paper, we define the following terms:

1. *State*: the analog, true position and time of something in the physical world.
2. *Map*: the discrete position-time values (called cells) at which NavQ predicts space vehicle (SV) availability and DOP.
3. *Grid*: a *map* with uniform spacing in each dimension, representing the world as discrete hyper-voxels.

We define the following variables:

- s: the number of SVs above the mask angle θ_{mask}.
- h: the *grid* of size M rows and N columns, representing the height of each 1×1 m tile.
- V: the total 4D hyper-volume which NavQ could output, defined by M rows, N columns, R altitude layers, and T time steps. M, N, and R will be at most the largest contiguous region inside the 3D city model used by NavQ, and T will be the maximum contiguous time of validity of the satellite almanac information used by NavQ.
- v: the 4D hyper-volume which the user chooses for NavQ to output, defined by m rows, n columns, h altitude layers, and t time steps. The user's input is valid if $v \in V$, and NavQ will output a map of shape v.

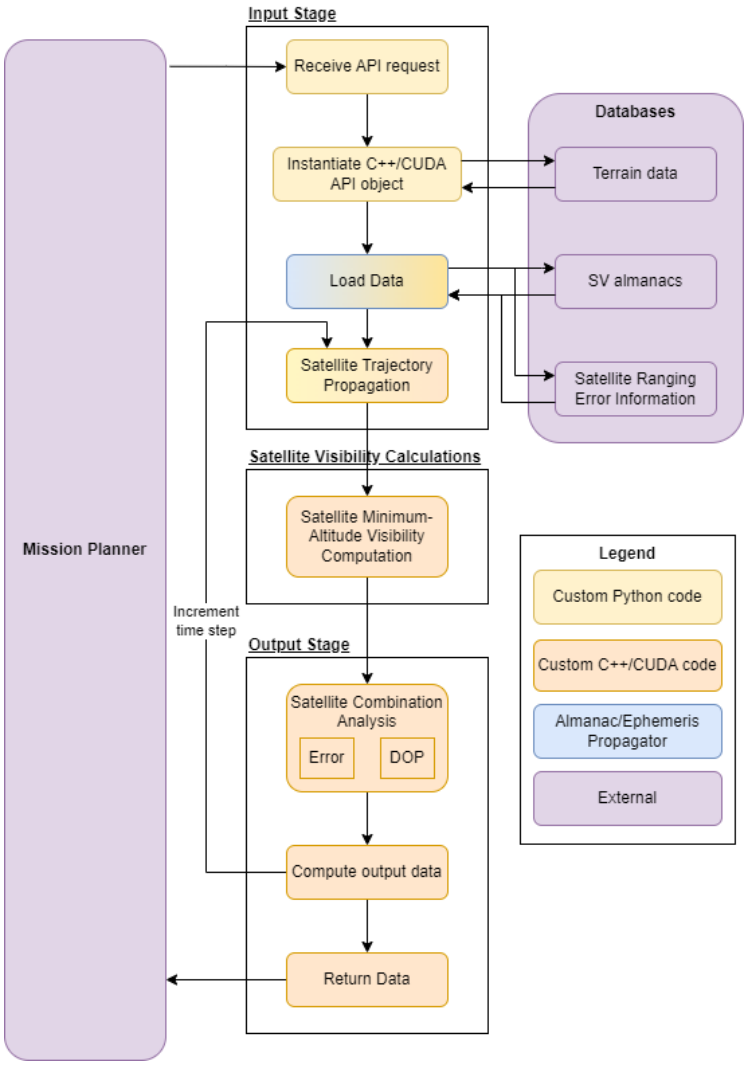


Figure 2: High-level block diagram of NavQ processing.

- h_{vis}: a 3D array with a size of M columns, N rows, and s satellites. This volume is used to indicate the height at which each satellite is visible at each point on the grid.
- h_{max}: the maximum height observed across the grid (this has a significant impact in how fast the algorithm can execute and will be discussed in more detail when addressing the stopping condition for the satellite visibility computation section).
- δ_{step}: the step size given by the mission planner. This value indicates the size of step the algorithm takes along the line defined by the SV position in the sky and the current position on the grid.
- d_{max}: the maximum distance needed to traverse along the line between the

satellite and the current point of interest for the algorithms described in the satellite visibility computation section.

3.2 Input Stage

The first stage of the framework reads and pre-processes the information used to compute satellite visibility calculator. The pre-processing steps discard information that has minimal impact on the final predictions and pre-compute intermediate information which is repeatedly accessed in the satellite visibility stage.

3.2.1 Inputs

NavQ requires three data product inputs: a 3D city model, a GNSS almanac, and satellite ranging error information. The user specifies the map within the regions included by the input data products.

3D City Model

For the 3D city model, the results in Section 4 Digital Surface Maps (DSMs) published by the National Oceanic and Atmospheric Administration (NOAA), such as the ones published in 2015 for Massachusetts, New Hampshire, and Rhode Island (, OCMP). These models are aerial LiDAR surveys with a 1×1 m resolution. The models are stored in GeoTIFF file format, containing a raster grid representing the DSM for the requested region, along with metadata indicating the map projection and affine transformation coefficients. This enables conversion to an earth-centered earth-fixed (ECEF) coordinate system.

Four notable trade-offs for this choice of city model are as follows. First, since the survey is an aerial LiDAR survey, only the maximum height within a 1×1 m region is sampled; the underside of overhangs are not represented, and complex structures like bridges are represented as a solid block. Second, since public-domain LiDAR datasets can be several years old, new buildings and changes to existing buildings may not be mapped. Third, variations in attenuation factors associated with the material makeup of obstructions are not considered. Lastly, foliage is oversimplified, and temporal changes (seasonal, growth, etc.) are not captured.

Depending on the method used to interpolate between samples in the DSM, this introduces a trade-off between under-reporting and over-reporting satellite visibility. For example, a min-pooling interpolation technique, which uses the height of the lowest LiDAR point as the reference height for the cell, is likely to over-report visibility, and a max-pooling interpolation technique, which chooses the height of the highest LiDAR point as the reference height for the cell, is likely to under-report visibility. Figure 3 depicts these interpolation methods. Similarly, treating all bridges and terrain surfaces as obstructions will, for most GPS performance analysis algorithms, minimize situations of over-reporting at the expense of under-reporting in some map cells. Section 4 compares the two approaches for an example city to demonstrate.

Additionally, the input DSM will impact the computation time of NavQ. The primary factors which are directly attributable to the input DSM construction are the size of the DSM, the maximum altitude observed within this area, and the variability of heights in the DSM. Thus, physical attributes of the city itself, such as

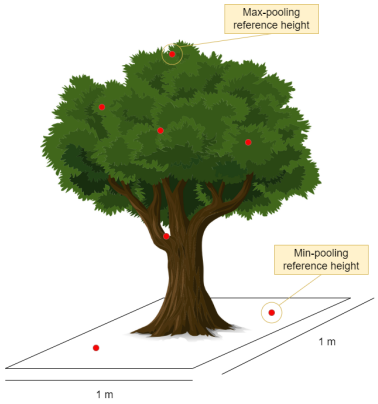


Figure 3: Example of 1×1 m tile with LiDAR points hitting a tree within it. Each red point represents a LiDAR data product and its corresponding height.

foliage and shapes of individual buildings, only impact the overall computation time through their variability on the scale of the resolution of the DSM.

There is also a computation time dependence on height that is due to the Traversal Stopping Condition in the parallelized LOS and shadow casting algorithms presented in Section 3.3. The LOS and shadow casting algorithms iterate longer when buildings are taller. For the LOS algorithm specifically, large variability in heights within the DSM will cause more shadows along these buildings. Alternatively, DSMs with low variability in the height of the buildings converge quicker, as individual points within the map will be processed faster.

In the results shown for this study, foliage is considered as opaque and unchanging. This is a simplification, since its impact on the model will be a function of season, weather, and 3D survey modality used. Foliage may attenuate, reflect, or absorb radio frequency (RF) transmissions from the survey equipment differently than it will in the satnav RF bands, leading to a discrepancy between 3D model and actual satnav signal propagation. A pair of studies conducted in forested areas with a mix of species (Moore et al., 2023b, 2022), showed that unless the depth of foliage between the SV and the receiver exceeds about 10 m, GNSS signals are attenuated but not totally blocked.

Almanac

Prediction time windows can be chosen using any of several almanacs which provide historical orbit data and several days of projected orbit data. NavQ interprets the SEM almanac format available from the United States Air Force GPS Operations Center (DoD, 2022). NavQ propagates the satellite states and stores the satellite elevation and azimuth relative to the central position-time value of the map for each timestamp.

Using the almanac may decrease prediction accuracy. However, the error introduced is negligible. For example, if the satellite positions have as much as 2 km of error (Misra and Enge, 2012), and this error lies purely in the worst-case dimension (perpendicular to the LOS between satellite and receiver), the radial error is approximately 0.0056°. Additionally, for computational simplicity, NavQ does not

correct for signal time-of-flight. Assuming that the satellite traverses 180° of the sky in 8 hours with a nominal distance of 20,200 km, the radial amount of change over the time-of-flight is 0.0005° per second. Even if these two sources of error are in the worst-case direction and aligned such that they compound, the total angular error is less than 0.01°.

Satellite Ranging Error Information

Satellite ranging errors may not be required in all cases, though some performance analysis algorithms may choose to include it. In this reference framework, an optional input is reserved for User Equivalent Ranging Error (UERE) information, represented as a data array where each entry represents each SV's ranging error in meters.

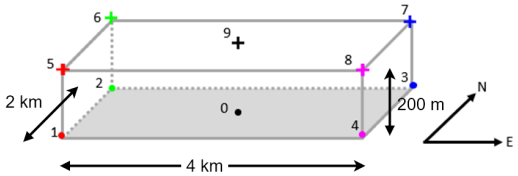
3.2.2 Input Pre-Processing

Since the scale of urban environments is small relative to a GNSS's RF signal path lengths, the input data can be decoupled from the forecast map cells with the assumption that the elevation and azimuth angle to each satellite is constant over the map cells. In the Satellite Trajectory Propagation (STP), each satellite modeled can be propagated only once per timestep (at the center of the map) and stored as an elevation angle and azimuth angle indexed by the SV ID and the time. The final list of SVs is filtered using a mask angle (θ_{mask}) set by the mission planner.

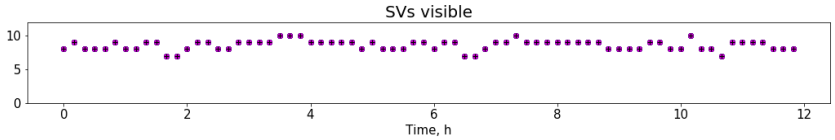
As Section 3.3 will describe, a GPU-accelerated function determines which SVs are visible from each point within the test volume. To enable parallelization in assessing the individual map cells, the algorithms are designed to find the minimum altitude value where each SV is visible within the map, given the obstructions of the terrain. Finally, this data is compressed, assuming that all given map cells with the same SV combination have the same performance metric score. This reduces the space of potential scores from a per-map-cell basis to a per-unique-SV-combination basis, making the computation cost of scoring an individual map cell as small as a lookup table. These values are saved, and the loop continues to the next time step.

To apply this assumption in a representative test case, consider the example problem of a terrestrial volume located in Boston, Massachusetts, USA, centered at 42.3665° N and 71.0719° W, spanning approximately 4 km east-to-west, 2 km north-to-south, and ranging vertically by 200 m. The extrema are illustrated in Figure 4(a) below. Propagating the SV trajectories over a 12-hour period beginning at UTC 2020-04-05 17:04:19 (the epoch time of an arbitrarily chosen SEM GPS almanac) and excluding SVs with a mask angle below 10° from calculations produces the variations in DOP captured in Figure 4.

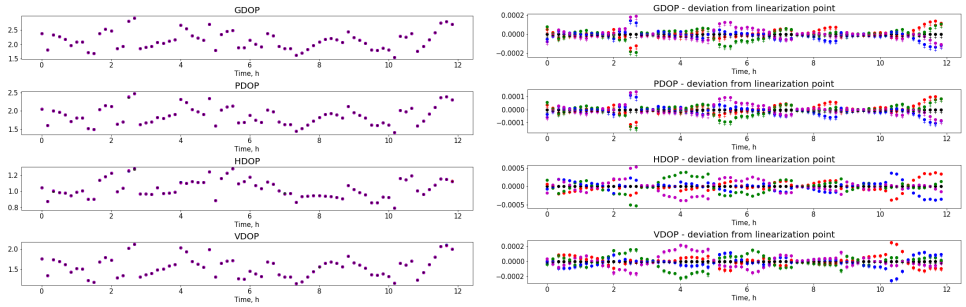
Figure 4 (b) plots the number of SVs visible for every point in Figure 4 (a). All extrema points in this volume saw the same combinations of satellites over the simulation period, as captured in Figure 4 (b). The variation in DOP even at opposite extremas (Figure 4 (d)) is in the order of a DOP factor of 0.001, several orders of magnitude smaller than the variation caused by changing SV geometry and visible combinations over the simulated interval (Figure 4 (c)).



(a) Definition of test volume extrema points used for the analysis.



(b) Number of SVs in LOS of the 10 selected extrema points.



(c) Data output values for the 10 selected extrema points. Note that the differences on the y axis are minuscule enough that points overlap at all time steps.

(d) Difference of navigation quality metrics between the reference point in the center of the map (black circle) and the 8 volume extrema points (colored circles or crosses).

Figure 4: DOP values plotted over 12 hour period for test volume points.

3.3 Satellite Visibility Calculations (SVC)

The second stage of the framework outputs the minimum altitude at which each SV is visible for a particular (r, c) map cell within the map's horizontal dimensions, where r represents the index along the row dimension, and c represents the index along the map's column dimension. This stage only contains one computational section which is named the "Satellite Minimum-Altitude Visibility Computation" (SMAV). Naïve or generalized implementations of this computation could be prohibitively time consuming. Thus, we propose an HPC-optimized, GNSS-intended shadow casting algorithm to improve the computation time burden. As a contrast, this section first introduces a similarly HPC-optimized LOS algorithm. Both approaches decouple the number of calculations needed from the map cell resolution in the altitude dimension. This way, the output stage can compare each altitude to the SV's minimum viewing altitude to determine if the SV will be visible, which may, by itself, improve computation time relative to black-boxed algorithms.

3.3.1 Line-of-Sight (LOS) Algorithm

The Line-of-Sight algorithm projects a line from a given horizontal coordinate and computes the minimum height necessary for LOS to the satellite. This algorithm is repeated for each SV (s) in the total number of available SVs (S) above the mask angle (θ_{mask}), and each point of interest (POI) at index $i = (r_i, c_i)$ within the $M \times N$ terrain grid and an SV with azimuth angle γ and elevation angle θ.

LOS Algorithm Definition

1. Initialize all values as if the SVs are all visible at the altitude of the POI on the grid:

$$h_{vis}[r_i][c_i][s] = h[r_i][c_i] \tag{1}$$

2. Calculate the maximum distance along the azimuth direction needed to check for SV visibility based on the mask angle, the highest altitude in the terrain grid, and the point altitude.

$$d_{max} = \frac{h_{max} - h[r_i][c_i]}{\tan(\theta_{mask})} \tag{2}$$

   Where d_{max} is the maximum distance needed to travel, h_{max} is the maximum height of the map, $h[r_i][c_i]$ is the height of the map cell at index i, and θ_{mask} is the mask angle provided by the mission planner.

3. Calculate the step sizes by which the ray trace increments the row and column indices, where δ_{step} is the step size given by the mission planner.

$$\begin{aligned} \Delta_r &= \sin(\gamma_s)\delta_{step} \\ \Delta_c &= \cos(\gamma_s)\delta_{step} \\ \Delta_d &= \sqrt{(\Delta_r)^2 + (\Delta_c)^2} = \delta_{step} \end{aligned} \tag{3}$$

   Δ_r represents the vertical step size (distance along the rows direction), Δ_c and represents the horizontal step size (distance moved along the columns direction). Note that the angle used to compute Δ_r and Δ_c is the SV azimuth angle γ, which means that the ray trace is observing values along the direction towards the satellite.

4. Initialize iteration variables:

$$\begin{aligned} r_{(k=1)} &= r_i + \Delta_r \\ c_{(k=1)} &= c_i + \Delta_c \\ d_{(k=1)} &= \Delta_d \end{aligned} \tag{4}$$

$d_{(k)}$ is increased after each step, given that the ray trace is moving in the direction of the satellite, where k represents the number of steps taken along the direction of the satellite.

5. Iterate along the SV azimuth vector points, while $(r_k, c_k) \in (M \times N)$ and $d_k < d_{max}$:

$$\begin{aligned} h_{vis,k} &= h[r_k][c_k] - d_k \tan(\theta_s) \\ h_{vis}[r_i][c_i][s] &= \max(h_{vis}[r_i][c_i][s], h_{vis,k}) \\ r_{(k+1)} &= r_k + \Delta_r \\ c_{(k+1)} &= c_k + \Delta_c \\ d_{(k+1)} &= d_k + \Delta_d \\ k &= k + 1 \end{aligned} \tag{5}$$

$h_{vis,k}$ represents the minimum altitude required for a viewer at POI i to see the SV s with an elevation angle of θ_s without being blocked by terrain at index k with altitude $h[r_k][c_k]$.

$h_{vis}[r_i][c_i][s]$ represents the highest known viewer altitude required to see the SV with an elevation angle of θ_s. It is updated with each iteration k so that at the end of the process, h_{vis} represents the viewer altitude needed to see the SV without being blocked by the highest obstacle along the azimuth angle γ_s of the SV.

Traversal stopping condition A stopping condition can be used to reduce the traversed distance along the azimuth angle vector by updating the d_{max} needed after each update to h_{vis}. This is possible because d_{max} represents the longest distance along any line for which the highest point in the grid would have an elevation angle equal to or greater than the receiver mask angle. If h_{vis} is updated, d_{max} can be recomputed to account for this new distance based on the new observed height and the elevation angle of the satellite. This can shorten the distance needed to find the maximum h_{vis}, especially when a large height value of h_k is measured. To implement this optimization, Equation 2 can be updated such that:

$$d_{max} = \frac{h_{max} - h_{vis}[r_i][c_i][s]}{\tan(\theta_s)} \tag{6}$$

Two particular optimizations for this application improve the computation time. First, due to the nature of the algorithm, the primary array h_{vis} does not need to be updated every iteration. For this reason, the value of h_{vis} can be stored in a register throughout the algorithm and only updated with the final value of h_{vis} at the end of the algorithm. This reduces the per-iteration computation cost of the traversal. Second, to avoid computationally-expensive division operations within a GPU kernel, the assumption of consistent satellite geometry across the position domain of the map allows for the inverse of the tangent of the SV elevation angle (needed for

the traversal stopping condition) to only be computed once at the beginning of the iteration. These two optimizations are specific to AAM/UAM GNSS visibility forecasting, and may not be present in naïve or generalized implementations.

3.3.2 Shadow Algorithm

In contrast to the LOS algorithm's receiver-centric view of the world, a shadow casting algorithm traces the shadows cast by buildings (as if each satellite is a sun) as a function of the SV elevation angles. The following algorithm is repeated for each SV (s) in S and each point of interest (POI) at index $i = (r_i, c_i)$ within the $M \times N$ terrain grid and an SV with azimuth angle γ and elevation angle θ.

Shadow Casting Algorithm Definition

1. Initialize all values as if the SVs are all visible at the altitude of the POI on the grid:

$$h_{vis}[r_i][c_i][s] = h[r_i][c_i] \tag{7}$$

2. Calculate the step size at which to increment the row and column indices, where δ_{step} is the step size given by the mission planner.

$$\begin{aligned} \Delta_r &= \sin(\gamma + \pi)\delta_{step} \\ \Delta_c &= \cos(\gamma + \pi)\delta_{step} \\ \Delta_h &= \tan(\theta)\delta_{step} \end{aligned} \tag{8}$$

   Δ_r represents the vertical step size (distance along the rows direction), Δ_c represents the horizontal step size (distance moved along the columns direction), and Δ_h represents the step size taken along the height of the *shadow* cast by objects within the map. Note that the angle used to compute Δ_r and Δ_c is the SV azimuth angle γ shifted by π. This shift is used because the algorithm moves in the opposite direction of the azimuth angle of the satellite.

3. Initialize iteration variables:

$$\begin{aligned} r_{(k=1)} &= r_i + \Delta_r \\ c_{(k=1)} &= c_i + \Delta_c \\ h_{(k=1)} &= h_i - \Delta_h \end{aligned} \tag{9}$$

   $h_{(k)}$ is decreased after each step, given that the cast is moving in the opposite direction of the satellite (towards the ground). This means the algorithm is casting the *shadow* of the current i position within the map, so the shadow decreases in altitude after each step.

4. Iterate along the opposite SV azimuth vector points, while $(r_k, c_k) \in (M \times N)$:

$$\begin{aligned} h_{vis}[r_k][c_k][s] &= \max(h_{vis}[r_k][c_k][s], h_k) \\ r_{(k+1)} &= r_k + \Delta_r \\ c_{(k+1)} &= c_k + \Delta_c \\ h_{(k+1)} &= h_k - \Delta_h \\ k &= k + 1 \end{aligned} \tag{10}$$

Following the opposite direction of the SV azimuth angle, a *shadow* is cast along this line to other grid points within the map by decreasing $h_{(k+1)}$. Figure 5 shows an example of the *shadow* line created by the building, where h_{vis} represents the minimum height at which the SV is visible. If the altitude at the grid point (r_k, c_k) is lower than the height of the *shadow* h_k, the cast continues with the next step along the shadow.

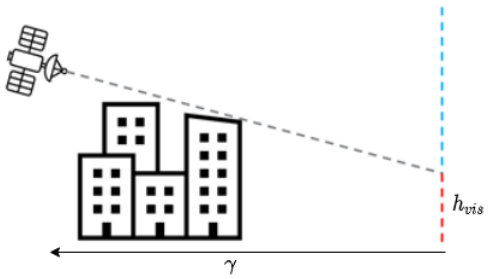


Figure 5: Line of sight to SV along azimuth direction γ. Note that h_{vis} represents the minimum distance above which the SV becomes visible. Not to scale.

Traversal stopping condition Step 4 iterates as long as the shadow remains within the map dimensions and the height of the *shadow* is above the minimum MSL altitude across the map. Additionally, the traversal can stop if the MSL altitude of the grid at point k along the shadow line is higher than the height of the *shadow*. This condition occurs only when the cast reaches the end of the shadow, allowing the algorithm to finish faster in most cases:

$$\begin{aligned} \textbf{if}(h_k <= h_{vis}[r_k][c_k][s]) \\ exit \end{aligned} \tag{11}$$

This exit has a significant performance impact, as the loop terminates sooner near low altitude-delta areas on the map (for example, wide-open spaces complete very quickly). Also, satellites with high elevation angles will complete the traversal quickly.

For computation reasons, GPU threads running the shadow casting algorithm should be spawned such that they index along the column first, then row, and then

the SV dimension. Indexing in the column dimension first will reduce memory latency in many HPC programming languages by allowing threads to make contiguous memory requests to the map and output arrays. Indexing in the SV dimension last will improve scheduling in many HPC programming languages by ensuring that threads modeling the different casts from a given SV execute together.

One constraint with this algorithm is that atomic operations are required on the update to the $h_{vis}[r_k][c_k][s]$ array, as the thread execution can happen out of order. For many HPC programming languages, this imposes a constraint on the type of memory that can store the output array.

Considerations for Future Work The framework described here could be adapted to support multipath modeling for error estimation. The multipath modeling step would logically be placed after the LOS computations described here. Future research will be needed to handle multipath modeling outputs efficiently and to determine whether the lookup tables described in Section 3.4 can be leveraged to handle generalized multipath results.

3.3.3 Performance Comparison

Choosing between the LOS and the shadow casting algorithm is primarily a function of the computing hardware available, as shown in Table 1. The shadow casting algorithm can outperform the LOS algorithm provided that the computing hardware can perform atomic operations efficiently, a requirement imposed by the updates to the h_{vis} array. Further, since the shadow casting algorithm requires more atomic writes for low-elevation satellites, there is a crossover point between the shadow and the LOS algorithm being more efficient, which is dependent on the atomic write speed of the specific hardware used. This requirement also makes implementing the shadow casting algorithm on a multi-GPU system much more difficult due to the memory synchronizations needed. However, the NavQ prototype demonstrates that satisfying these requirements is reasonable with modern consumer-grade GPU hardware. Section 4 will show that, when using appropriate GPU hardware, the shadow casting algorithm offers a noticeable computation time improvement with only a minimal difference in prediction accuracy.

NavQ performs its vector rasterization (the vector along the SV line) using the digital differential analyzer (DDA) algorithm. This choice is captured in the formulation of the LOS and shadow casting algorithms' method for updating the steps along the rows and columns in Equation 5 and Equation 10. This method does require a proper selection of δ_{step} to ensure a proper line traversal, as shown in (Gutierrez et al., 2022). (Gutierrez et al., 2022) also showed that a step size of 0.1 m can guarantee a 95% coverage of the grid cells touched by the line, but the performance suffers compared to a step size of 1.0 m. Due to this, NavQ implements the custom DDA-Dynamic line method extension tested by Gutierrez *et al.* to guarantee 100% line coverage without the significant performance degradation from using a small step size. DDA-Dynamic computes the size of the step needed to take to reach the next grid cell. It is more computationally intensive to run each iteration, but it significantly drops the total number of iterations.

Table 1: Table of the differences between LOS algorithm and Shadow algorithm.

Algorithm	LOS	Shadow Casting
GPU work division	1 thread per grid point per SV	
Memory updates	1 element in the array	all elements in the array along the shadow
Atomic operations	No	Yes
Stopping Condition	Maximum distance traveled given by the tallest building on the map	Casted shadow is lower than previous shadow/building
Speed	Slower than Shadow	Faster than LOS
Multi-GPU	Easy	Hard

3.4 Output Stage

The output stage is responsible for converting the lower-dimensionality satellite availability calculations into a four-dimensional navigation risk-correlated map. Under the assumption that the SV geometry for a given timestep is consistent across the urban area being modeled, there exist a finite number of possible DOP values on the map. Thus, instead of computing the DOP at each cell on the map, NavQ instead computes DOP for all possible satellite visibility combinations. This ensures that no computation is repeated, makes the output map generation a lookup operation, and guarantees fewer operations overall so long as at least two cells share the same satellite visibility combination.

3.4.1 Satellite Combination Analysis (SCA)

This stage converts the h_{vis} array of size $M \times N \times S$ of SV visibility altitudes, where each layer in S corresponds to the visibility altitudes of each SV above the horizon, into a bitmap representing the combination of SVs visible at each map cell. Based on the assumption that no more than 32 SVs will be above the mask angle, these combinations are represented with 32-bit integers. Each bit represents whether the corresponding satellite in the satellite list is visible. Next, these combinations are all filtered into a *unique* list of keys. These 32-bit keys are used in a key-value pair to store the final data outputs in key-indexed arrays. The following steps detail the logic used to obtain the final output data.

Combination Algorithm

1. Sort the SVs by the height at which they are visible for each grid point. A conventional bubble sort algorithm is sufficient for this task, since the size of the sort is at most 32.

2. Compute the bitmap representation of the combination of SVs in LOS using the sorted array. Each 32-bit integer is generated by taking a bit-wise sum of binary integers, where a single bit 1 is shifted k digits to represent the kth SV being visible. For instance, if SVs 0, 2, and 3 are visible, the integer representing the combination is obtained by bit-wise summing the numbers 0001, 0100, and 1000. This combination yields a value of 1101 in binary or 13 in base 10.

3. Create a list of unique key combinations.

   (a) Sort all of the keys in the entire array. This sort is much larger, and requires a GPU-optimized sort such as those found in the CUDA Thrust library (Bell and Hoberock, 2012).

   (b) Compute the *unique* list of keys and how many times they appear in the array using a reduce-by-key operation[1].

4. For each unique key combination, compute the key-indexed table outputs such as the HDOP, PDOP, GDOP, the error covariance matrix, and the error ellipse formed by the horizontal components of the error covariance matrix. DOP computations are performed only for combinations with more than four SVs (from any GNSS constellations, under the assumption that the receiver correctly accounts for time reference differences between constellations) and less than 20 SVs (see discussion in Section 4.1).

While the GPU is better suited for most of the computations performed in this problem, Step 4 is an exception. This step is dominated by generating and applying performance metric operations on a relatively small number of satellite geometry matrices of small dimensionality relative to the GPU thread block size. While GPUs are well-known to outperform CPUs on large matrix multiplies, CPUs can be preferable to GPUs for small matrices when other operations are queued, as even launching the minimum number of GPU threads on an architecture may still leave many threads idle with no work to do. Further, more sophisticated performance modeling algorithms are likely to demand additional memory for intermediate calculations. The available memory per thread is typically much smaller on a GPU than on a CPU. So, once on the CPU, a thread is launched for each unique SV combination key. This CPU thread computes the H matrix. The CPU also performs the operations to obtain the covariance matrix and all of the outputs required in Step 4 of the Combination Algorithm before copying the results back to the GPU. With Step 4 complete, there is now a representation of the GPS performance modeling map in memory, condensed as a key-sorted array indexed by the satellite availability combination. The final data output index tables are copied to the GPU to be referenced in the final map indexed by position-time cell.

This combination representation is inherently non-linear. One should take care when modeling multiple constellations to ensure that the computational burden of

[1]This operation takes in an array and outputs a smaller array with the *unique* list of keys and optionally a second array indicating how many times each key appeared on the original array. Sorting is necessary to avoid duplicate keys and to simplify key searches.

this step does not grow to an impractical cost. Most urban centers are located at latitudes with approximately 30 or fewer satellites above the horizon simultaneously when using all of today's operational satnav systems. As Section 4.1 will show, this is roughly the upper limit of what is computationally practical without extraordinary computing resources. Additionally, cells with this many LOS satellites should contain the states with the lowest AAM/UAM navigation risk. If there is a need to model more satellites, other assumptions will likely be needed to reduce the complexity again.

3.4.2 Compute Output Data and Return Data

When data is requested from the service for a given hyper-volume $v(m \times n \times h \times t) \in V(M \times N \times \mathbb{R} \times T)$, a function is used to retrieve the data for each individual hyper-voxel $v_{i,j,k,l} \in (m \times n \times h \times t)$. This function first retrieves the key combination from the sorted array h_{vis} by iterating through each height at which a new satellite is visible until the height is larger than the height for each POI $v_{i,j,k,l}$. At this point, the key for this hyper-voxel is the SV combination observed at the previous height.

Once the key has been found, this key is then used to determine the index in the data output tables. A binary search algorithm is performed between the key and the values stored in the sorted key array. Once the index where the key and the key table match is found, this index is used to reference the output data table to retrieve the corresponding value requested by the mission planner. These operations are all executed within the same kernel on the GPU.

The data output tables store the value for HDOP, PDOP, GDOP, SV count, and the SV combination on the GPU. The covariance matrix and the horizontal position error distribution ellipse are stored on the CPU.

4 Results and Field Comparison

NavQ was prototyped and tested on a system with two Intel Xeon Silver 4216 CPUs, each with 16 cores/32 threads (hyperthreading enabled), 128 GB of DDR4 ECC RAM and an A6000 NVIDIA GPU. A Python wrapper performs high-level setup and lightweight calculations (e.g. satellite propagation), while the geometry calculations in Figure 2 are written in C++ and CUDA. This section analyzes the resultant prototype's computation cost and accuracy. First, Section 4.1 studies the time taken per step in the framework and discusses how design parameters impact the performance. Next, Section 4.2 compares NavQ's satellite availability predictions to data collected in the field and discusses how assumptions in the design lead to different accuracy and types of errors.

4.1 Computational Performance Analysis

To assess the computational performance, this section profiles NavQ's computation time on a custom scenario designed to be computationally taxing for the Satellite Visibility Calculations stage of NavQ. The scenario chosen for this analysis places 15 satellites at equidistant azimuth angles, and 15° elevation angles with respect to

the center of the map, as depicted in Figure 6. This geometry includes the typical number of satellites visible for GPS+GLONASS fused navigation, ensures that most grid points can see many satellites, and requires more iterations from the LOS and Shadow algorithms than is expected in realistic scenarios. For the input terrain, this scenario uses an 8.03 km^2 DSM of downtown Boston, Massachusetts, USA. The selected area of Boston is characterized by relatively consistent street-level altitude, but significant variability in building height. Although the buildings are generally rectangular, the large variation in height means that some DSM points will always be tall relative to their neighbors, and that "wide-open" stretches will not exist for long. As described in Section 3.3, these are the conditions that require more iterative steps before the traversal stopping condition is met.

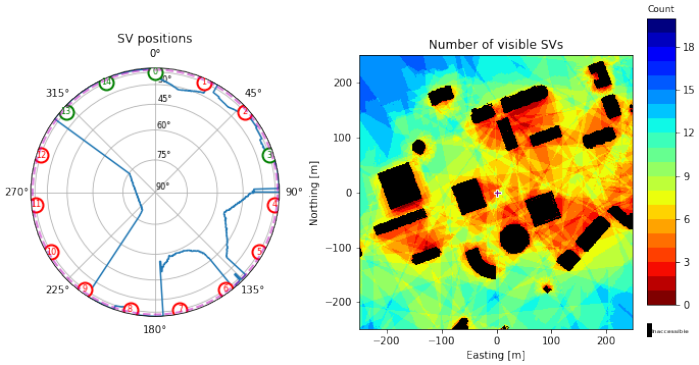


Figure 6: Example skyplot of the satellite locations and the visibility at a height of 40 m within an area of Boston, MA, USA.

Table 2 captures NavQ's processing time for this scenario with varying step sizes and satellite visibility calculation methods. The reported times are an average of five runs in each configuration. Given the size of the input DSM and the number of satellites, the GPU launched a total of 120 million threads for the Satellite Minimum Altitude Visibility step. The Compute Output step fetches a 2 km $\times$ 2 km surface indicating the height at which a minimum of four satellites are visible. This data is a representative use case for data that a mission planner can request to evaluate the safety of a path taken by a vehicle within this urban scenario.

In addition to the order-of-magnitude computation time metrics offered, Table 2 confirms some of the tradeoffs discussed in Section 3. First, the shadow casting algorithm consistently outperforms the LOS algorithm in the Satellite Minimum Altitude Visibility step regardless of the line method and step size used, as anticipated for this single-GPU architecture with efficient atomic operations. Second, varying the step size shows the shadow casting algorithm's dependence on atomic operations, as a smaller step size requires more atomic operations, which reduces the performance difference between the two algorithms. Third, since DDA-Dynamic is faster than DDA with a step size of 0.5m for both the LOS and shadow casting algorithms, and since DDA-Dynamic can achieve 100% accuracy for line coverage, DDA-Dynamic proves to be the superior option provided that the threads processing it has sufficient resources to perform its increased computation requirements.

Table 2: Average execution time for the each algorithm stage with 15 equally spaced SVs at a 15-degree elevation angle for the different line methods and algorithms developed.

Algorithm	**Line Method**	**Step Size [m]**	**STP stage [s]**	**SMAV stage [s]**	**SCA stage [s]**	**Compute Output [s]**
Line of Sight	DDA	0.1	0.01	1.72	0.07	0.02
		0.5		0.38		
		1		0.22		
	DDA-Dynamic	N/A		0.33		
Shadow	DDA	0.1		1.58		
		0.5		0.28		
		1		0.13		
	DDA-Dynamic	N/A		0.2		

Next, Table 3 shows how NavQ's performance varies as a function of the number of modeled satellites. The results in this table are from the Shadow algorithm with DDA-Dynamic, but the following conclusions are general to any of the configurations of Table 2. As expected, the only step which scales non-linearly with respect to the number of satellites N_{sats} is the Satellite Combination Analysis step. While the number of unique IDs will always be less than or equal to $2^{N_{sats}}$, the trial with $N_{sats} = 20$ shows that the service will still model the majority of the possible combinations, but this effect decreases with $N_{sats} > 20$. This performance degradation is the conclusion presented in Section 3, that the largest practical value for near real-time performance for N_{sats} is 20 (without extraordinary computing hardware). A value of N_{sats} larger than this would increase the Number of Unique IDs and, as a result, increase the Satellite Combination Analysis stage execution time beyond 1 s.

Table 3: Average execution time for each stage of the algorithm with different equally-spaced SVs at a 15-degree elevation angle, for the Shadow algorithm with DDA-Dynamic method.

Number of Satellites (N_{sats})	**Number of Unique IDs**	**STP Stage [s]**	**SMAV Stage [s]**	**SCA Stage [s]**	**Compute Output [s]**
10	1023	0.01	0.13	0.03	0.02
15	32,767		0.2	0.06	
20	821,749		0.25	0.47	
25	5,851,303		0.33	3.39	
30	17,476,500		0.39	10.75	

Lastly, when implementing a framework that targets GPUs, the developer needs to consider the available resources at a macro and micro scale. From the macro scale, RAM and data bandwidth are significant factors when prototyping NavQ. In

our scenarios, urban maps often required at least 4GB of RAM for a single timestep, making this tool unsuitable for lower-tier GPUs. After choosing the A6000 GPU, the additional bandwidth from the GDDR6 DRAM technology proved particularly valuable by keeping individual memory fetches and writes short. From the micro scale, it is essential to consider bottlenecks such as GPU utilization, branch divergence, register allocation per thread, and shared memory.

4.2 Satellite Availability Forecasting Analysis

To evaluate the composite effect from the GNSS assumptions chosen in the design of NavQ, this section compares the satellite visibility predictions from the NavQ prototype to the observed visible satellites by a vehicle traveling through the urban canyons of Corpus Christi, Texas, USA. The exact numerical performance will vary based on the scenario, but the following discussion presents the generalizable findings.

For NavQ, an aerial LiDAR scan conducted for the United States Geological Survey (USGS) in 2018 with an average point spacing of 0.6 m was used to generate the DSM input for the region of Corpus Christi, Texas, USA. This region is characterized by a moderately-sized urban environment, where the variability in building height is less than that of the Boston DSM used above. Blockage by trees is unlikely. The plants along this trajectory are predominantly decorative bushes and shrubs; the vehicle never passes near thick foliage that is taller than the antenna mount. The Corpus Christi LiDAR survey was converted into two separate DSMs with 1×1 m cell resolution, one DSM generated by max-pooling the survey data, and one by min-pooling the survey data. The mask angle used was 15°.

Truth Reference For an empirical reference, two receivers recorded data from a roof mount on an automobile driving a roughly four-mile path over approximately one hour on June 6, 2022. The path started and ended at a low-density region, traversed through an unobstructed area near the bayfront and coursed into the urban canyons of Corpus Christi's city center, which is depicted in Figure 7. The two receivers were:

1. A u-Blox ZED-F9P receiver configured to receive L1 and L2 band signals from GPS and GLONASS.

2. A VectorNav VN-200 GNSS-aided Inertial Navigation System (INS) configured to receive L1 signals from GPS and produce a tightly integrated navigation solution. This offered a cross-reference to the ZED-F9P.

The ZED-F9P's observables were post-processed in RTKLIB demo5 (version 2.4.3/demo5/b31) (Everett, 2024) to remove signals that did not meet a validity criterion (carrier-to-noise ratio greater than 10dB, signal observed within one second, and no cycle slips). The demo5 variant of RTKLIB (Takasu et al., 2007) is intended to operate on measurements from lower-cost receivers in degraded signal conditions. This enables the use of measurements from the ZED-F9P which would otherwise have been filtered out by the strict signal quality selection criteria in conventional RTKLIB. The ground truth used to match the path to corresponding grid

cells in NavQ was produced by applying a temporal smoothing of the receiver output via the u-Blox u-Center utility for the lateral motion. The reference altitude for the ground truth was set to 2 m above ground level of the DSM.

This results in conservative decisions on whether a received signal is LOS. For example, if the received signal is dominated by an LOS component, but if a secondary multipath component is also present, RTKLIB demo5 is likely to reject it, resulting in that SV being classified as blocked for that state. Similarly, if the receiver is tracking a particularly strong multipath reflection with sufficient time delay relative to the LOS signal, and RTKLIB demo5 correctly detects that the receiver is tracking a multipath reflection, the SV would again be classified as blocked for that state. In both cases, it would appear that NavQ is overestimating that SV's availability for the corresponding grid cell. Directional antennas, received signal polarization analysis, and peak shape analysis may be necessary to improve the accuracy of the empirical reference data.

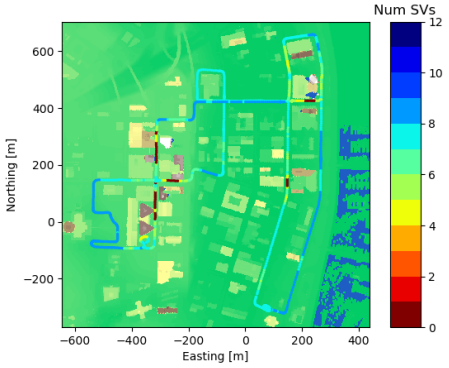
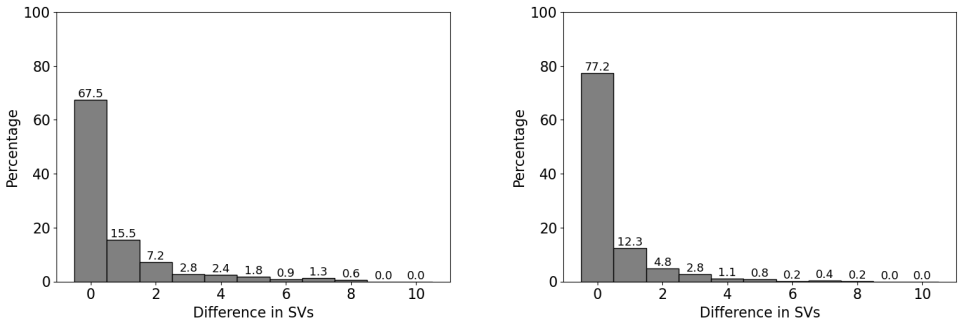


Figure 7: Measurement drive path from June 6, 2022 superimposed on DSM of Corpus Christi, Texas. The path color corresponds to the number of predicted LOS SVs (scale at right) at each of the 2898 measurement times. In the urban canyons at the left portion of the path, navigation quality is severely degraded by urban terrain.

Accuracy Analysis Figure 8 summarizes the differences between the reported SVs in LOS by the simulator and those reported by the post-processed measurements. These results are an expansion from the analysis in our companion conference paper (Moore et al., 2023a). These expanded results compare the SV IDs reported by the simulator and RTKLIB. For the max-pooling DSM, NavQ's predictions and the post-processed measurements reported the same combination of LOS satellites 67.5% of the time. This accuracy increased to 77.2% for the min-pooling DSM configuration.

To better understand how NavQ and the post-processed results differ, Figure 9 compares the max-pooling DSM and the min-pooling DSM through a percentage confusion matrix of the satellite count from the post-processed results versus NavQ. For the analysis of these confusion matrices, we define the following terms:

1. A Type I error is defined as a false positive incident, meaning that NavQ reports more visible satellites than the post-processed results (upper-right triangle in



(a) Max-pooling DSM (b) Min-pooling DSM

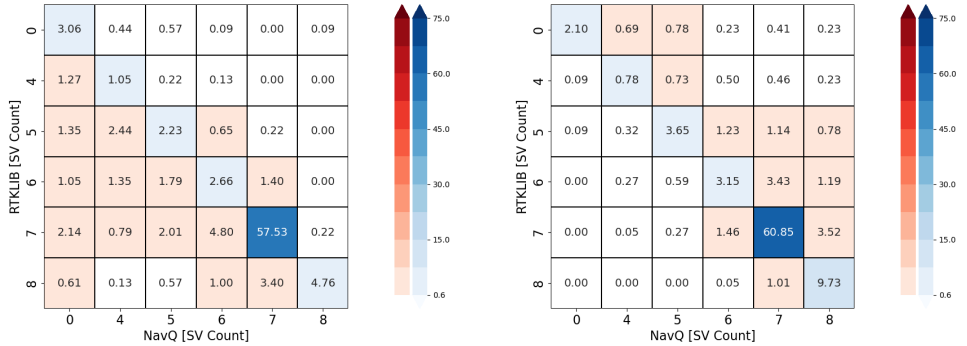
Figure 8: Histogram of the differences of SVs in LOS between the empirical truth reference and the simulator using the max-pooling DSM (a) and the min-pooling DSM (b).

the confusion matrix). These errors are typically more problematic than false negatives (Type II) in the AAM/UAM flight planning perspective since they will overestimate the "goodness" of a grid cell. Additionally, we define Critical Type I errors as the Type I incidents that happen when the post-processed results reported less than four visible satellites. We consider these critical since a flight planner using the prognostic service could designate such a cell as viable or even good while, in actuality, no navigation solution consisting exclusively of LOS signals is possible without some form of aiding.

2. Type II errors, or false negatives, are the incidents where the post-processed results report more visible satellites than NavQ. These are generally more tolerable in AAM/UAM contexts for the reasons described above.

The blue cells along the diagonals are the percentage of the measurements where both post-processed measurements and NavQ report the same SV count. The lower left triangle of cells under the diagonal represented the percentage when NavQ reported less visible satellites than the post-processed measurements (Type II errors). The upper right triangle of cells over the diagonal were the measurements when NavQ reported more visible satellites than the post-processed measurements (Type I errors). The top row cells, excluding the cell on the diagonal, are the errors where NavQ predicts a viable navigation solution when none is possible without aiding (Critical Type I errors).

Table 4 sums cells of the same type from the max-pooling and min-pooling confusion matrices in Figure 9. Min-pooling achieves exact matches to the empirical reference dataset over 80% of the time and meets the requirement of over 95% accuracy within two SVs. Max-pooling has lower overall accuracy, consistent with the description in Section 3.2. Max-pooling achieves just over 70% exact matches and just over 90% accurate within two SVs but also has a much lower percentage of Type I. This offers a trade-off: max-pooling is a better option if the AAM/UAM mission planner prefers to minimize the Type I errors, particularly the Critical Type I



(a) Max-pooling DSM

(b) Min-pooling DSM

Figure 9: Confusion matrix of the SV count reported by RTKLIB and the simulator using max-pooling DSM (a) and min-pooling DSM (b). Each matrix cell is labeled with its percentage prevalence and is colored per the scale at the right.

errors.

This trade-off is also dependent on the urban environment itself. Maxpooling is expected to perform better when the majority of structures have edge lengths on the order of integer multiples of NavQ's grid size, as the majority of the hyper-volume of each cell will typically be full. Similarly, minpooling is expected to perform better around more jagged, spare urban structures, where the majority of the hyper-volume of each cell would be empty. Alongside our related observations in Section 3.2, studying the DSM construction as a function of prediction accuracy is an interesting topic for future study.

Table 4: Type I and Type II error comparison between RTKLIB and NavQ using max-pooling and min-pooling DSMs.

Metric	**NavQ with max-pooling DSM**	**NavQ with min-pooling DSM**
Type I errors	4.02%	15.53%
Critical type I errors	1.18%	2.33%
Type II errors	24.71%	4.20%
Exact Matches	71.28%	80.26%
Correct within 2 SVs	90.92%	95.97%

So, given this distinct shift between Type I and Type II errors from max-pooling and min-pooling, DSM resolution stands out as a critical factor in overall accuracy. If the DSM resolution is increased by an order of magnitude, the impact on the overall accuracy would likely be larger compared to some of the other assumptions (such as applying the satellite geometry at the center of the map to all cells in the grid or using the almanac instead of the ephemeris) were eliminated. However, this

increase in resolution would also come with an increased computation cost.

These tests were also conducted by comparing the LOS and Shadow algorithms with the different line methods. Regardless of the line method or algorithm used for the Satellite Altitude Visibility segment, the resulting differences were negligible (± 2 mismatches) and, thus, not elaborated on here. Since the differences between these algorithms are negligible, the assumptions made for the Shadow line algorithm are viable.

5 Conclusions and Future Work

Today, regulatory authorities and the research community are intensively evaluating GNSS-based navigation as a critical technology for emerging AAM/UAM operations. This work aimed to stimulate further study of this specific field by presenting an HPC-accelerated shadow casting algorithm, alongside an HPC-accelerated conventional LOS algorithm. The computation time required to generate an AAM/UAM navigation risk-correlated map with a prototype simulator based on the shadow casting approach was detailed. In addition to prediction errors arising from to NLOS/multipath signals, an assessment of the composite prediction accuracy identified the DSM data as the primary driver of error.

Further experimental validation is underway to compare NavQ with real-time field measurements. In its current form, NavQ has been used at NASA and other agencies as a pre-operations flight tool. Further design effort is underway to include modeling of some NLOS/multipath signals and techniques for bounding the GNSS receiver error.

Appendix

Computing DOP

To compute an instantaneous position solution, conventional GPS receivers first measure the range to each visible satellite, then iteratively refine an initial guess of the receiver's position and time using a linearized system of equations. This multilateration problem can be represented as:

$$\mathbf{H}\boldsymbol{\Delta}\mathbf{x} = \boldsymbol{\Delta}\rho \tag{12}$$

Here, $\mathbf{H}$ is the observation matrix, formed from the unit vector to the satellites in the local navigation frame; $\boldsymbol{\Delta}\mathbf{x}$ is the state vector representing the receiver position and the receiver clock bias; and $\boldsymbol{\Delta}\rho$ encompasses the pseudorange measurements to the satellites.

Assuming that pseudorange measurement errors are zero-mean Gaussian with the same variance σ^2_{URE}, the relationship between position error covariance $\mathbf{cov}(d\mathbf{x})$ and pseudorange error covariance $\mathbf{cov}(d\rho)$ is given by:

$$\mathbf{cov}\left(d\mathbf{x}\right) = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{cov}\left(d\rho\right) = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\left(I_{nxn}\sigma^2_{URE}\right) \tag{13}$$

Readers are advised to consult (Kaplan and Hegarty, 2017) for more details about the GPS solution and the derivation of its covariance matrix.

Following the equivalent-variance Gaussian pseudorange assumption stated above, Equation 13 relates the satellite geometry matrix to the covariance in the receiver's state estimate:

$$\begin{aligned}\mathbf{cov}\left(d\mathbf{x}\right) = &\begin{bmatrix} \sigma_x^2 & cov(x,y) & cov(x,z) & cov(x,c\delta t) \\ cov(x,y) & \sigma_y^2 & cov(y,z) & cov(y,c\delta t) \\ cov(x,z) & cov(y,z) & \sigma_z^2 & cov(z,c\delta t) \\ cov(x,c\delta t) & cov(y,c\delta t) & cov(z,c\delta t) & \sigma_{c\delta t}^2 \end{bmatrix} \\ = &\sigma_{URE}^2 \begin{bmatrix} D_{11} & D_{12} & D_{13} & D_{14} \\ D_{21} & D_{22} & D_{23} & D_{24} \\ D_{31} & D_{32} & D_{33} & D_{34} \\ D_{41} & D_{42} & D_{43} & D_{44} \end{bmatrix}\end{aligned} \tag{14}$$

This shows that the coefficients of the DOP matrix $\left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-\mathbf{1}}$ describe the magnitude of the receiver's error covariance ellipsoid in each dimension of the receiver's state. If the position components of $d\mathbf{x}$ are expressed in the east-north-up (ENU) coordinate frame, the DOP matrix can be dissected into four sub-metrics which are easy to interpret from a human perspective and are easy to integrate with policy or other sensors.

$$\begin{aligned} GDOP &= \sqrt{D_{\mathbf{11}} + D_{\mathbf{22}} + D_{\mathbf{33}} + D_{\mathbf{44}}} \\ PDOP &= \sqrt{D_{\mathbf{11}} + D_{\mathbf{22}} + D_{\mathbf{33}}} \\ HDOP &= \sqrt{D_{\mathbf{11}} + D_{\mathbf{22}}} \\ VDOP &= \sqrt{D_{\mathbf{33}}} \end{aligned} \tag{15}$$

GDOP refers to the geometric dilution of precision and represents the ratio of position and timing error to the range error along all dimensions, or in other words, how the error in the measurement will affect the final location estimate. PDOP or positional (3D) dilution of precision refers to the positional elements of the DOP matrix. Horizontal dilution of precision (HDOP) represents the 2D positional elements of the DOP matrix, i.e., positional error in the east-north plane. Lastly, vertical dilution of precision (VDOP) encompasses the vertical (up) element of the DOP matrix. A lower DOP value is considered a better solution, and values under ten are often considered geometries that may provide sufficient positional accuracy for many applications.

Equations 13 and 14 show that these metrics depend on two factors, the range error and the satellite geometry. The range error factor σ_{URE}^2 is the model of error variance due to signal characteristics, orbit estimation errors, and receiver characteristics. This factor is typically modeled for short time and space intervals from field measurements. The geometry factor (DOP), alternately, is a result of the satellite geometry only, and can be predicted for any location using the satnav system's broadcasted satellite location data, such as the almanac or ephemeris of GPS.

Additional information can be determined from the position error covariance matrix $\mathbf{cov}\,(d\mathbf{x})$. As the derivation in Equation 14 is expressed in a local navigation frame, the diagonal elements of the position error covariance matrix $\mathbf{cov}\,(d\mathbf{x})$ represent variances along the directions of the axes of the local coordinate system. However, the principal components of the position error distribution are not necessarily aligned with the local navigation frame. Suppose the errors are assumed to be Gaussian. In that case, the 3-dimensional position error distribution is ellipsoidal. The ellipse's primary axes are defined by the eigenvectors of the error covariance matrix, and the corresponding eigenvalues define their respective magnitudes.

As our primary concern for this effort is horizontal positioning error, covariance analysis is only considered along the horizontal portion of the position error distribution. This is achieved by considering only the upper-left 2x2 sub-matrix of the covariance matrix, referred to here as matrix $\mathbf{A}$.

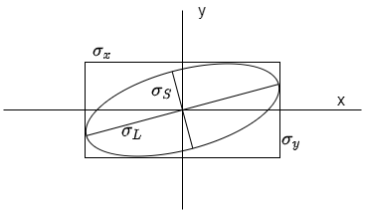


Figure 10: The horizontal position error distribution defined by the principal components σ_L and σ_S is bounded by the rectangle defined by σ_x and σ_y. Based on image from (Kaplan and Hegarty, 2017).

The horizontal position error distribution may be visualized as an ellipse whose boundary is defined by the 1σ distance from its center, as shown in Figure 10. Note that the ellipse is bounded by a box defined by σ_x and σ_y. The vectors representing the distribution's principal components form the ellipse's semi-major and semi-minor axes. These vectors are determined from the $2x2$ position covariance matrix's eigenvectors e_1 and e_2 and eigenvalues λ_1 and λ_2 (Johnson and Wichern, 2014):

$$\begin{aligned} |\mathbf{A} - \lambda\mathbf{I}| &= 0 \\ (\mathbf{A} - \lambda_j\mathbf{I})\mathbf{e}_j &= 0, \text{ for j = 1, 2} \end{aligned} \tag{16}$$

Here, vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ are the unit vectors defining the principal components of the 2-dimensional distribution defined in the local frame, and $\sqrt{\lambda_1}$ and $\sqrt{\lambda_2}$ are their corresponding magnitudes. These are shown as σ_L and σ_S in Figure 10. The angle of the horizontal position error distribution's primary axis can be determined by:

$$\theta = \mathbf{atan2}(\mathbf{e}_{1y}, \mathbf{e}_{1x}) \tag{17}$$

In future work, this 2-dimensional error distribution and its directionality could be used to calculate navigation safety metrics based on the directional spread of the error distribution and its proximity to obstacles or other hazardous zones. For the

time being, HDOP is considered a sufficient metric for error estimation, as discussed in Section 2.

References

Ali, K., Chen, X., Dovis, F., De Castro, D., and Fernández, A. J. (2012). Multipath estimation in urban environments from joint GNSS receivers and LiDAR sensors. *Sensors (Basel, Switzerland)*.

Anyaegbu, E. and Hansen, P. (2022). GNSS performance evaluation for deep urban environments using GNSS Foresight. In *Proceedings of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022)*, pages 1127–1136.

Appleget, A. and Bartone, C. (2019). A consolidated GNSS multipath analysis considering modern GNSS signals, antenna, installation, and boundary conditions. In *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, pages 2855–2869.

Bell, N. and Hoberock, J. (2012). Thrust: A productivity-oriented library for cuda. In *GPU computing gems Jade edition*, pages 359–371. Elsevier.

Ben-Moshe, B., Elkin, E., Levi, H., and Weissman, A. (2011). Improving accuracy of GNSS devices in urban canyons. *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry, CCCG 2011*.

Betz, J. W. (2015). *Engineering satellite-based navigation and timing: global navigation satellite systems, signals, and receivers*. John Wiley & Sons.

Bhamidipati, S., Kousik, S., and Gao, G. (2021). Set-valued shadow matching using zonotopes. In *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, pages 2373–2390.

Bijjahalli, S., Gardi, A., and Sabatini, R. (2018). GNSS performance modelling for positioning and navigation in urban environments. In *2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pages 521–526.

Bourdeau, A., Sahmoudi, M., and Tourneret, J.-Y. (2013). Prediction of GNSS signal bias using a 3D model in urban environments. In *Proc. European Navigation Conference (ENC 2012)*.

Bradbury, J. (2007). Prediction of urban GNSS availability and signal degradation using virtual reality city models. In *Proceedings of the 20th international technical meeting of the satellite division of the Institute of Navigation (ION GNSS 2007)*, pages 2696–2706.

Causa, F. and Fasano, G. (2021). Multiple UAVs trajectory generation and waypoint assignment in urban environment based on dop maps. *Aerospace Science and Technology*, 110:106507.

Costa, E. (2011). Simulation of the effects of different urban environments on GPS performance using digital elevation models and building databases. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):819–829.

Dill, E., Gutierrez, J., Young, S., Moore, A., Scholz, A., Bates, E., Schmitt, K., and Doughty, J. (2021). A predictive GNSS performance monitor for autonomous air vehicles in urban environments. In *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, pages 125–137.

DoD (2022). Global positioning system operations center.

Drevelle, V. and Bonnifait, P. (2011). Global Positioning in Urban Areas with 3-D Maps. In *IEEE Intelligent Vehicles Symposium (IV 2011)*, pages 764–769.

Dunn, M. J. (2022). NAVSTAR GPS space segment/navigation user segment interfaces (IS-GPS-200), revision N. Technical report, Space Systems Command (United States Space Force), https://www.gps.gov/ technical/icwg/IS-GPS-200N.pdf.

Everett, T. (2024). RTKLIB demo5. Accessed: February 1, 2024.

Furukawa, R., Kubo, N., and El-Mowafy, A. (2020). Prediction of RTK-GNSS performance in urban environments using a 3D model and continuous LOS method. In *Proceedings of the 2020 International Technical Meeting of The Institute of Navigation*, pages 763–771.

Goodrich, K. H. and Theodore, C. R. (2021). Description of the NASA urban air mobility maturity level (UML) scale. In *AIAA SciTech 2021 Forum*, page 1627.

Groves, P. D., Wang, L., Adjrad, M., and Ellul, C. (2015). GNSS shadow matching: The challenges ahead. In *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2015)*, pages 2421–2443.

Gutierrez, J., Kaeli, D., Dill, E. T., and Closas, P. (2022). Performance and accuracy assessment of line marching algorithm computations utilizing GPUs within a predictive GNSS quality service. In *AIAA SciTech 2022 Forum*, page 0029.

Jakobsen, J. and Jensen, A. (2013). Simulating non-LOS GNSS reflected signals in urban and dense urban environments. In *Proceedings of the 26th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2013)*, pages 1670–1674.

Ji, S., Chen, W., Ding, X., Chen, Y., Zhao, C., and Hu, C. (2010). Potential benefits of GPS/GLONASS/GALILEO integration in an urban canyon–Hong Kong. *The Journal of Navigation*, 63(4):681–693.

Johnson, R. A. and Wichern, D. W. (2014). *Applied multivariate statistical analysis*, volume 6. Pearson London, UK:.

Johnson, W. and Silva, C. (2022). NASA concept vehicles and the engineering of advanced air mobility aircraft. *The Aeronautical Journal*, 126(1295):59–91.

Kaplan, E. and Hegarty, C. (2017). *Understanding GPS/GNSS Principles and Applications*. Artech house.

Kbayer, N. and Sahmoudi, M. (2018). Performances analysis of GNSS NLOS bias correction in urban environment using a three-dimensional city model and GNSS simulator. *IEEE Transactions on Aerospace and Electronic Systems*, 54(4):1799–1814.

Kleijer, F., Odijk, D., and Verbree, E. (2009). Prediction of GNSS availability and accuracy in urban environments case study Schiphol airport. *Location based services and telecartography II: from sensor fusion to context models*, pages 387–406.

Lee, Y.-W., Suh, Y., and Shibasaki, R. (2008). A GIS-based simulation to predict GPS availability along the Tehran Road in Seoul, Korea. *KSCE Journal of Civil Engineering*, 12:401–408.

Misra, P. and Enge, P. (2012). *Global Positioning System: Signals, Measurements, and Performance*. Ganga-Jamuna Press, 2 edition.

Moore, A., Gutierrez, J., Dill, E., Logan, M. J., Glover, S., Young, S., and Hoege, N. (2023a). Accuracy assessment of two GPS fidelity prediction services in urban terrain. In *2023 IEEE/ION Position, Location and Navigation Symposium (PLANS)*.

Moore, A., Rymer, N., Glover, J. S., and Ozturk, D. (2023b). Predicting GPS fidelity in heavily forested areas. In *2023 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 772–780. IEEE.

Moore, A., Schubert, M., Rymer, N., Villalobos, D., Glover, J., Ozturk, D., and Dill, E. (2022). Volume raycasting of GNSS signals through ground structure lidar for UAV navigational guidance and safety estimation. *AIAA SciTech Forum*.

Nagai, K., Fasoro, T., Spenko, M., Henderson, R., and Pervan, B. (2020a). Evaluating GNSS navigation availability in 3-D mapped urban environments. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 639–646.

Nagai, K., Fasoro, T., Spenko, M., Henderson, R., and Pervan, B. (2020b). Evaluating GNSS navigation availability in 3-D mapped urban environments. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 639–646.

Neamati, D., Gupta, S., Partha, M., and Gao, G. (2023). Neural city maps for GNSS NLOS prediction. In *Proceedings of the 36th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023)*, pages 2073–2087.

Obst, M. (2014). *Bayesian Approach for Reliable GNSS-based Vehicle Localization in Urban Areas*. Doctoral thesis, Chemnitz University of Technology.

(OCMP), O. P. (2015). 2013-2014 U.S. Geological Survey CMGP LIDAR: Post Sandy (MA, NH, RI).

O'Connor, M., Ruwisch, F., Kersten, T., Skupin, C., Ren§, L., Wuebbena, T., and Schön, S. (2021). Low-latency GNSS multipath simulator for real-time applications in autonomous driving. In *2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–9.

Patterson, M. D., Isaacson, D. R., Mendonca, N. L., Neogi, N. A., Goodrich, K. H., Metcalfe, M., Bastedo, B., Metts, C., Hill, B. P., DeCarme, D., Griffin, C., and Wiggins, S. (2021). An initial concept for intermediate-state, passenger-carrying urban air mobility operations. In *AIAA SciTech 2021 Forum*, page 1626.

Pongsakornsathien, N., Bijjahalli, S., Gardi, A., Symons, A., Xi, Y., Sabatini, R., and Kistan, T. (2020). A performance-based airspace model for unmanned aircraft systems traffic management. *Aerospace*, 7(11):154.

Sanromà Sánchez, J., Gerhmann, A., Thevenon, P., Brocard, P., Ben Afia, A., and Julien, O. (2016). Use of a fisheye camera for GNSS NLOS exclusion and characterization in urban environments. In *Proceedings of the 2016 International Technical Meeting of The Institute of Navigation*.

Smolyakov, I., Rezaee, M., and Langley, R. B. (2020). Resilient multipath prediction and detection architecture for low-cost navigation in challenging urban areas. *Navigation*, 67(2):397–409.

Suh, Y., Konishi, Y., Hakamata, T., Manandhar, D., Shibasaki, R., and Kubo, N. (2004). Evaluation of multipath error and signal propagation in complex 3D urban environments for GPS multipath identification. In *Proceedings of the 17th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2004)*, pages 1147–1156.

Tadic, S., Moura, G., and Trichaud, T. (2013). Design of GNSS performance analysis and simulation tools in the "gapfiller" web portal. In *2013 21st Telecommunications Forum Telfor (TELFOR)*, pages 236–239.

Takasu, T., Kubo, N., and Yasuda, A. (2007). Development, evaluation and application of RTKLIB: A program library for RTK-GPS. In *GPS/GNSS symposium*, pages 213–218.

Tokura, H. and Kubo, N. (2016). Effective satellite selection methods for RTK-GNSS NLOS exclusion in dense urban environments. In *Proceedings of the 29th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2016)*.

van Diggelen, F. (2021). End game for urban GNSS: Google's use of 3D building models. *Inside GNSS*, 16:42–49.

Zhong, Q. and Groves, P. (2023). Optimizing LOS/NLOS modeling and solution determination for 3D-mapping-aided GNSS positioning. In *Proceedings of the 36th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023)*, pages 373–402.

Ziedan, N. I. (2017). Urban positioning accuracy enhancement utilizing 3D buildings model and accelerated ray tracing algorithm. In *Proceedings of the 30th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2017)*, pages 3253–3268.