High-Fidelity Simulation of Horizon-Based Optical Navigation with Open-Source Software

Shimane, Yuri; Miraldo, Pedro; Berntorp, Karl; Greiff, Marcus; Elango, Purnanand; Weiss, Avishai

TR2023-128 October 03, 2023

Abstract

In this work, a high-fidelity simulation environment for autonomous cislunar spacecraft navigation based on horizon-based optical navigation (OpNav) is developed. Spacecraft autonomy is becoming a critical need, both from a mission safety standpoint in case of communication failure with Earth, as well as a scalability standpoint as the number of deployed spacecraft increases dramatically in the coming years. Autonomy is of particular importance in deep space, where communication (and thus, navigation) relies on NASA’s Deep Space Network (DSN), which will need to be rationed, or where links to DSN can fail (see, for example, the recent CAPSTONE mission). As spacecraft activity builds on the near rectilinear halo orbit (NRHO) that the Lunar Gateway will fly, autonomous navigation and station-keeping control methods that do not rely on DSN are required. This work details how a high-fidelity rendering-based simulation environment for horizon-based OpNav may be realized. While the environment is useful for any spacecraft transiting in the vicinity of the Moon, special considerations arising from conducting OpNav on an NRHO are highlighted. The proposed environment is tested by processing rendered images of the Moon along an NRHO using a state-of-the-art horizon-based OpNav technique, and insights on the accuracy of the obtainable position measurements are provided.

International Astronautical Congress (IAC) 2023
HIGH-FIDELITY SIMULATION OF HORIZON-BASED OPTICAL NAVIGATION WITH OPEN-SOURCE SOFTWARE

Yuri Shimane¹, Pedro Miraldo³, Karl Berntorp³, Marcus Greiff³, Purnanand Elango², and Avishai Weiss³*

¹Georgia Institute of Technology, USA, yuri.shimane@gatech.edu
²University of Washington, USA, pelango@uw.edu
³Mitsubishi Electric Research Laboratories, USA, {miraldo,berntorp,greiff,weiss}@merl.com

Abstract

In this work, a high-fidelity simulation environment for autonomous cislunar spacecraft navigation based on horizon-based optical navigation (OpNav) is developed. Spacecraft autonomy is becoming a critical need, both from a mission safety standpoint in case of communication failure with Earth, as well as a scalability standpoint as the number of deployed spacecraft increases dramatically in the coming years. Autonomy is of particular importance in deep space, where communication (and thus, navigation) relies on NASA’s Deep Space Network (DSN), which will need to be rationed, or where links to DSN can fail (see, for example, the recent CAPSTONE mission). As spacecraft activity builds on the near rectilinear halo orbit (NRHO) that the Lunar Gateway will fly, autonomous navigation and station-keeping control methods that do not rely on DSN are required. This work details how a high-fidelity rendering-based simulation environment for horizon-based OpNav may be realized. While the environment is useful for any spacecraft transiting in the vicinity of the Moon, special considerations arising from conducting OpNav on an NRHO are highlighted. The proposed environment is tested by processing rendered images of the Moon along an NRHO using a state-of-the-art horizon-based OpNav technique, and insights on the accuracy of the obtainable position measurements are provided.

Keywords: Optical navigation, NRHO, Autonomy

1. Introduction

The Moon has seen a resurgence of activity, with nation states and private enterprise sending, or planning to send, orbiters and landers, and deploying human outposts, e.g. the Lunar Orbital Platform-Gateway (LOP-G)¹², as part of a renewed interest in exploring and, eventually, settling the Moon. This increased activity will stress the capacity of Deep Space Network (DSN)-class ground-based infrastructure that is used to provide communication and navigation to cislunar spacecraft, motivating the search for alternative solutions. Furthermore, as recently highlighted by the CAPSTONE mission³, links to DSN may fail⁴, jeopardizing mission safety. While the CAPSTONE spacecraft eventually regained communication ability⁵, such events demonstrate the need for independent autonomous navigation solutions. Towards this end, the Lunar GNSS Receiver Experiment (LuGRE) will attempt to use GPS and Galileo signals in the lunar environment and on the lunar surface⁶, and CAPSTONE is testing a new peer-to-peer autonomous navigation system that measures range and range-rate relative to NASA’s Lunar Reconnaissance Orbiter (LRO)³.

An alternative navigation approach is to autonomously process space-captured optical imagery. The history of optical navigation (OpNav) with humans in the loop dates back to the early days of spaceflight in the mid 20th century, when Apollo astronauts used nautical techniques, much like traditional mariners, to take sightings with a sextant⁷⁸. Space-based optical navigation has come a long way, with terrain-relative and proximity operation applications seeing increased usage, and the recent success of the DART Mission, which used the DRACO (Didymos Reconnaissance and Asteroid Camera for OpNav) telescope to image, navigate to, and collide with the asteroid Dimorphos⁹. Of interest in this work, horizon-based OpNav technology has recently matured, and was demonstrated by the Orion

¹Yuri interned at Mitsubishi Electric Research Laboratories during the development of this work.
*Corresponding Author.
Multi-Purpose Crew Vehicle on the Artemis 1 mission [10,11]. Horizon-based OpNav uses the horizon, or lit limb, of a celestial body to provide information about the relative position of the observer with respect to the observed celestial body [12]. Combining optical observations of a celestial body with independent attitude information readily available, e.g., from a star tracker, yields inertial position measurements that may be used in a navigation filter to provide full spacecraft state estimation.

Many of the characteristics of optical-based measurements can only be well-understood through realistic imagery, such as the impact of terrain features, and the diminishing observability as a function of range from the observed celestial body in the camera boresight, amongst other effects. However, the difficulty in procuring a robust set of lunar images from verified true positions, motivates the generation of synthetic imagery. To date, several works have explored the use of computer generated imagery to simulate OpNav. NASA’s EDGE software [13] was used to generate synthetic imagery of the Moon as part of the validation of the Orion optical navigation system [14]. Vizard, which is part of the Basilisk software [15], is a Unity-based astrodynamics visualization library, that has been used for horizon-based OpNav in Mars [16] and near rectilinear halo orbit (NRHO) [17]. POV-Ray [18] is a ray-tracing program that has been used by multiple authors, including for Moon-relative horizon-based OpNav [19,20], and as well as terrain-relative applications [21]. Blender is an open-source rendering engine with an active development community, and has previously been used for OpNav application in proximity operations [22], small-body-relative motion [23], as well as for terrain-relative applications [24]. Finally, Unreal Engine, a game engine from Epic Games, has also been explored for its usability in space-based applications, close to terrain [25], and for proximity operations [26,27].

In this work, a high-fidelity simulation environment for horizon-based OpNav is created using Blender. The steps involved in this development, from the selection of the rendering software, to the various considerations regarding the rendering process that impact the performance of OpNav, are discussed. This environment is used to render images of the Moon as seen by a fixed field of view camera, traveling along a baseline NRHO propagated in a full-ephemeris model. We highlight NRHO-specific considerations when using OpNav, particularly due to the large variation in perilune and apolune distances, as well as the geometric orientation of the orbit with respect to the Sun. The resulting images are processed by the non-iterative horizon-based OpNav algorithm from Christian and Robinson [28] to obtain position vector measurements. Insights on the impact of the various physical constraints as well, as engineering choices on the accuracy of the measurements are provided.

The remainder of this paper is organized as follows: first, in Section 2, we provide an overview of the full-ephemeris dynamical model that is used to study NRHOS, as well as a brief discussion on NRHOS themselves. This is followed by Section 3 where we detail the Blender-based simulation setup that generates images for horizon-based OpNav. In Section 4, the horizon-based OpNav algorithm is reviewed, and considerations specific to employing this technique for navigation along an NRHO are discussed. Numerical results based on rendered images and the implemented OpNav algorithm are provided in Section 5. Finally, Section 6 provides concluding remarks.

2. Spacecraft Model

This section provides an overview of the full-ephemeris-based dynamical spacecraft model and introduces near rectilinear halo orbits.

2.1 Equations of Motion

Consider a spacecraft in cislunar space under the gravitational influence of the Moon, Earth, and Sun, a Moon-centered inertial frame $F_I$, and a lunar principal axes frame $F_P$. The equations of motion for the spacecraft are given by

$$\dot{r} = v,$$

$$\dot{v} = -\frac{\mu}{r^3}r + a_{J2} + \sum_i a_N + a_{SRP},$$

where $r$ is the position of the spacecraft with respect to an unforced particle collocated with the center of the Moon, $v \equiv \dot{r}$ is the velocity of the spacecraft with respect to $F_I$, $\mu$ is the standard gravitational parameter of the Moon, $r = ||r||$, $a_{J2}$ is the perturbing acceleration due to the dominant coefficient in the lunar spherical harmonic gravitational model, $a_N$ is the acceleration on the spacecraft due to third-body $i \in \{\text{Earth}, \text{Sun}\}$, and $a_{SRP}$ is the perturbing acceleration due to solar radiation pressure (SRP). The accelerations in (1) represent the major predictable external forces acting on the spacecraft [29], and are
given by

\[
\mathbf{a}_{J2} = -\frac{\mu}{r^3} T^3_P \begin{bmatrix}
  \frac{3}{2} J_2 \left( \frac{R}{r} \right)^2 \left( 1 - \frac{5 z^2}{r^2} \right) \\
  \frac{3}{2} J_2 \left( \frac{R}{r} \right)^2 \left( 1 - \frac{5 z^2}{r^2} \right) \\
  \frac{3}{2} J_2 \left( \frac{R}{r} \right)^2 \left( 3 - \frac{5 z^2}{r^2} \right)
\end{bmatrix},
\]  

(2)

where \(\mathbf{r}|_P = [x, y, z]^T\) is the spacecraft’s position vector resolved in \(F_P\), \(T^3_P \in \mathbb{R}^{3 \times 3}\) is the rotation matrix that transforms components of a vector resolved in \(F_P\) into the components of the same vector resolved in \(F_I\), \(R\) is the reference equatorial radius of the Moon, and \(J_2\) is the coefficient of the \(J_2\) term;

\[
\mathbf{a}_{N_i} = -\mu_i \left[ \frac{r_i}{r_i^3} + \frac{d_i}{d_i^3} \right],
\]  

(3)

where \(d_i\) is the position of perturbing body \(i\) with respect to the Moon, \(d_i = \|d_i\|\), and \(r_i = r - d_i\) is the position of the spacecraft with respect to perturbing body \(i\);

\[
\mathbf{a}_{SRP} = P_\odot \left( \frac{1\text{AU}}{r_\odot} \right)^2 C_r A \frac{r_\odot}{m} r_\odot,
\]  

(4)

where \(r_\odot = r - d_\odot\) is the position of the spacecraft with respect to the Sun, \(d_\odot\) is the position of the Sun with respect to the Moon, \(r_\odot = \|r_\odot\|\), \(P_\odot\) is the SRP magnitude at 1 AU, \(C_r\) is the radiation pressure coefficient, \(A\) is the Sun-lit surface area of the spacecraft, and \(m\) is the spacecraft mass.

Table I summarizes the constant parameters of the dynamics, along with parameters of the SRP term for a hypothetical spacecraft.

### 2.2 Near Rectilinear Halo Orbit

When studying spacecraft motion in cislunar space, simplified dynamical models are often employed to obtain initial guesses for possible motions; one such model is the circular-restricted three-body problem (CR3BP), which assumes only the gravitational influences of the Earth and the Moon, corotating about their common barycenter on respective circular orbits. In the CR3BP, numerous families of periodic orbits may be numerically constructed; the NRHO of interest in this work is a member of the halo orbit family about the L2 point with a period corresponding to a 9:2 resonance with respect to the synodic period, where the spacecraft completes 9 revolutions for every 2 synodic periods, exhibiting nearly stable characteristics within the simplified CR3BP model.

While the analogous NRHO in the full-ephemeris model becomes only quasi-periodic, the orbit still offers relatively high stability. In addition, the orbit offers nearly eclipse-free motion, where the spacecraft remains outside of the Earth’s penumbral shadow for most of the time. These features make this NRHO well-suited for the crewed space station in cislunar space. Figure 1 shows the trajectory of the full-ephemeris NRHO in the Earth-Moon rotating frame and the inertial J2000 frame.
Table 1: Constants in full-ephemeris dynamics

<table>
<thead>
<tr>
<th>Dynamics parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moon gravitational parameter $\mu$, km$^3$/s$^2$</td>
<td>4902.80582147760</td>
</tr>
<tr>
<td>Earth gravitational parameter $\mu_\oplus$, km$^3$/s$^2$</td>
<td>398600.432896939</td>
</tr>
<tr>
<td>Sun gravitational parameter $\mu_\odot$, km$^3$/s$^2$</td>
<td>132712197035.766</td>
</tr>
<tr>
<td>Moon equatorial radius $R$, km</td>
<td>1737.4</td>
</tr>
<tr>
<td>Moon $J_2$</td>
<td>0.0002024</td>
</tr>
<tr>
<td>Astronomical unit, km</td>
<td>149597870.7</td>
</tr>
<tr>
<td>SRP magnitude at 1 AU $P_\odot$, N/m$^2$</td>
<td>4.56e-6</td>
</tr>
<tr>
<td>Radiation pressure coefficient $C_r$</td>
<td>1.15</td>
</tr>
<tr>
<td>SRP ballistic coefficient $m/A$, kg/m$^2$</td>
<td>500</td>
</tr>
</tbody>
</table>

3. Simulation of Optical Measurement

The development process for a simulation environment of optical measurements is discussed in this section. We first provide an overview of the various software that were considered. Then, considerations for rendering images for the purpose of OpNav are highlighted; specifically, spacecraft navigation requires uniquely precise measurements, and care must be taken to properly configure the rendering software. This is followed by a description on the modeling of the imaged celestial body, sunlight, and any eclipsing body. Finally, piecing everything together, we provide a procedural guide on how such an environment may be used to generate horizon-based optical measurements.

3.1 Comparison of Rendering Engines

There are multiple rendering software that may be used for the purpose of simulating optical navigation. At the early stages of this work, we tested using either Blender or Unreal Engine 5 (UE5), and concluded that Blender was better-suited for the task at hand. We avoided POV-Ray due to the inactive development community. While the Basilisk/Vizard suite provides a ready-made set of functions for this purpose, we also opted against this option due to our need for customizability and control over the rendering process.

When making the selection of rendering software, its interoperability with generic programming languages is critical, as it facilitates the simulation process that involves not only rendering images but also processing them and computing quantities that may rely on generic scientific routines. For this purpose, both Blender and UE5 have interfaces with Python, through which SPICE [31,32] routines, which is a system of data files and functions developed by the Navigation and Ancillary Information Facility at NASA’s Planetary Science Division for space mission modeling, planning, and execution, as well as other third-party scientific libraries, may be called.

A fundamental difference between Blender and UE5 is the primary purpose of the respective software; Blender is a rendering engine, whose purpose is to render scenes accurately, while UE5 is a game engine, whose purpose is to rapidly render interactive scenes. Thus, while Blender has numerous settings that may improve the physical accuracy of the ray-tracing process at the cost of longer rendering times, UE5 has more emphasis on clever approximations that can be made to speed up the rendering of a scene that looks realistic to the human eye. While UE5’s fast rendering speed is attractive for simulation, this comes at a cost of poor representation of edges, which is not visible to the naked eye by design, but becomes apparent when zooming in closely.

This is visible in Figure 2, which shows two example frames of the Moon rendered using Blender, Vizard, and Unreal Engine 5. The third row shows a zoomed-in view of the limb of the images in the second row. While both Blender and Vizard have a staircase pattern along the zoomed-in limb, shown on the bottom left and bottom center windows, the rendered image from UE5 contains irregular artifact pixels along the limb, as shown on the bottom right window. This effect led us to choose Blender over UE5 for the remainder of this work in order to be able to evaluate OpNav performance on NRHO under a well-understood simulation environment. Nevertheless, we note that while unnatural, the artifact from the UE5 image could also be interpreted as mimicking real image effects, where various types of noise in the image sensor can creep in. While this is beyond the scope of this work, it may also be reasonable to adopt UE5 for the simulation and study OpNav using
these imperfect, arguably corrupted images.

3.2 Rendering Considerations

In order to simulate measurements for horizon-based OpNav, it is critical that the rendered image contains the lit limb of the imaged body at the physically accurate location. Caution is required when using rendering engines that are targeted for real-time applications, such as video games, as the Level of Details (LOD) setting on the software is typically tuned to sacrifice the physical accuracy of objects that are far from the camera (thus occupying fewer pixels within the image) in favor of rendering speed. This is visible in Figure 2 from the bottom right window, corresponding to the zoomed-in view of the lit limb from UE5. While the bottom left and bottom center windows, corresponding to the zoomed-in view of the lit limb from Blender and Unity, have a crisp “staircase” pattern along the limb, the image from UE5 has irregular rugged pixels along the limb, which is unnatural given that these images are rendered without terrain features.

In addition to LOD settings, two more considerations relating to rendering have an important influence on the accuracy of the generated measurements. The first is the level of anti-aliasing applied. Aliasing refers to the artifacts caused by the rendering process, which results in jagged lines being formed. In Blender, this is controlled via the setting on pixel filtering [33]. Figure 3 shows the effect of the value of pixel filtering used when rendering the image of the Moon. While for the sake of horizon-based OpNav, a jagged limb is preferable, a “realistic” setting would be dependent on what the rendered image should simulate. A typical digital camera would have an anti-aliasing filter built-in, and therefore images produced by an off-the-shelf camera are likely to include anti-aliasing. On the other hand, if the camera is specifically configured for the purpose of OpNav, it is possible to assume that the anti-aliasing filter is removed, or the process is reversed through deconvolution. In this work, under the assumption that the spacecraft is equipped with an OpNav-purposed camera, the anti-aliasing is removed, by setting the pixel filtering to 0.

The second is on the ceiling or flooring operation which the rendering software takes to determine whether a sufficient portion of the pixel is above a threshold value. Depending on the convention used by the software, the ceiling or flooring operation results in some pixels at the edge of the physical limb being falsely coloured, which averages out to the limb being shifted by 0.5 pixels, either towards the top left or the bottom right of the image. This effect is illustrated in Figure 4, taking as an example the case where the rendering software rounds up any pixel that contains a portion of the Moon’s lit limb within it. Supposing that the pixel is not illuminated regardless of whether < 50% or > 50% of the pixel should be physically occupied by the lit limb, it is possible to see that the location of the actual limb would be portrayed in the image with a −0.5 pixel bias shift to the right. Hence, in this case, it would be appropriate to shift back any detected limb point by +0.5 pixel.

3.3 Components in Simulation Environment

In a minimal optical navigation simulation, the observed body and sunlight must be included. To augment the fidelity of the simulation, it is possible to consider the inclusion of the terrain of the observed body, along with any other neighboring celestial body that may cast a shadow on the observed body.

3.3.1 Modelling the Observed Celestial Body

Horizon-based OpNav algorithms work with ellipsoidal models for the observed body, which can be easily generated in rendering software. In the case of the Moon, since the reference ellipsoidal model as defined by the Planetary Constants Kernel (PCK) from SPICE has equal principal axes, it can be modelled as a sphere. In Blender, one option is to use a UV sphere, with a sufficient number of segments in the horizontal and vertical directions; in this work, 200 segments (vertical segments) and 100 rings (horizontal segments) are used. In addition, the “shade smooth” feature is applied to “change the way the shading is calculated across the surfaces (normals will be interpolated), giving the illusion of a smooth surface” [34].

To increase the fidelity of the simulation, the lunar terrain is introduced as a displacement of the mesh’s surface with respect to the surface normal. In Blender, this is achieved by using the Displacement Node to the material of the mesh. The displacement can be computed from an elevation map of the celestial body; in this work, the displacement of the Moon obtained from the Lunar Orbiter Laser Altimeter (LOLA) instrument, provided by NASA, with a 64 pixels per degree resolution, is used [35].

The inclusion of terrain is particularly important when simulating horizon-based OpNav of the Moon; as reported by Christian [36], when a sub-pixel level edge finding technique (to be discussed in Section
Fig. 2: Rendering comparison of the Moon. All images are taken using 2048 × 2048 pixels, and a field of view of 56.66° for the images in the first row, and 13.77° for images in the second row. The third row corresponds to a zoomed-in view on the top of the limb on the images in the second row. All images here are rendered without terrestrial features.

Pixel filter: 0.0  
Pixel filter: 0.5  
Pixel filter: 1.0

Fig. 3: Effect of pixel filtering on the lit limb of the Moon within the rendered image.

4.2) is employed, the local terrain of the Moon becomes important as soon as the Moon’s radius corresponds to about 48 pixels within the image. A strong proponent of this effect is the non-negligible discrepancy between the center of mass of the Moon and the geometric center of a triaxial ellipsoidal approximation of it; because the Moon is tidally locked, the far side of the Moon has experienced a significantly greater number of impact events, resulting in a far more rugged terrain than the near side. Since the OpNav algorithm relies on a triaxial ellipsoidal assumption of the observed celestial body, deviations of the best-fitting ellipsoid have a direct impact on the algorithm’s performance. While beyond the scope of this work, one may consider compensating for this deviation by incorporating corrective terms on the geometric center, dimensions, and orientation of the best-fitting ellipsoid, as reported in [37].
Fig. 4: Illustration of ceiling/flooring operation taken by software when rendering pixels on the edge of the lit limb. Ideally, the pixel with the limb occupying < 50% should not be illuminated, while the pixel with the limb occupying ≥ 50% should be.

3.3.2 Modelling Sunlight

In typical spacecraft OpNav applications, due to the much larger distance to the Sun compared to the distance from the observed body to the spacecraft, sunlight can be well-approximated using a directional light source. This is preferable to placing a point light source, which can be prohibitive due to the difference of length scales that are involved. For example, in the cislunar case, the Sun-Moon distance is almost $10^5$ times the radius of the Moon, making it impractical to place a point light source far into the distance while also having a sufficiently detailed mesh model for the Moon.

In reality, the Sun is a light source of finite volume. In Blender, the Sun Light object offers an angle variable, which dictates the angular diameter of the light source [38]. This becomes particularly important when modeling eclipses.

3.3.3 Inclusion of Eclipsing body

When the Moon is within the shadow cast by the Earth, the Moon’s limb becomes partially or entirely invisible to the spacecraft camera. While a total lunar eclipse typically lasts only about 30 minutes to an hour, the eclipse itself may last for a few hours. While its impact may be negligible on the long-term OpNav simulation of a cislunar spacecraft, it can be included to add fidelity to the simulation of operating a spacecraft around the epoch of a lunar eclipse. Figure 5 shows rendered images during the lunar eclipse on 2025-03-14 UTC on the top row, along with rendered images without the Earth (and therefore without a lunar eclipse) for comparison on the bottom row. During this eclipse, the Moon enters the Earth’s penumbra at 03:57:28 UTC, the partial eclipse begins at 05:09:40 UTC, and the full eclipse begins at 06:26:06 UTC.

3.4 Simulation Process

When simulating optical measurements from a spacecraft, any arbitrary reference frame may be used to locate the spacecraft, sunlight, as well as any relevant celestial bodies. In the case of simulating images for horizon-based OpNav, a practical reference frame to use is the principal axes frame of the primary celestial body that is being imaged. By using the principal axes frame, the mesh of the primary celestial body does not need to be moved or rotated, thus simplifying the simulation process.

Using the principal axes frame, the high-level process for generating the optical measurement can be summarized by the following steps:

1. Create mesh of the primary celestial body, with its principal axes aligned with the $xyz$-directions of the rendering environment;
2. Query position of the Sun $r_\odot$ in the principal axes frame from SPICE ephemeris data;
3. Create directional light with angle based on $r_\odot$;
4. Query position of any additional celestial bodies that may cast a shadow on the imaged body from SPICE and place a mesh at this position;
5. Place the camera’s position and orientation based on the spacecraft’s position and orientation.

Once a rendering environment is initialized, a new measurement may be obtained by going through steps 2 through 5.
4. Horizon-Based Optical Navigation on NRHO

Various state estimation problems can be posed in the context of horizon-based OpNav, depending on the unknowns \[12\]. In this work, we consider the problem with known spacecraft attitude and unknown spacecraft position. This is a relevant and realistic scenario when considering autonomous spacecraft navigation, since conventional sensors, such as star trackers, can provide attitude information to high precision \[12, 39\]. Specifically, we consider the Christian-Robinson algorithm \[28\] due to its noniterative, high-performing characteristics, as previously reported in the literature \[12, 28, 36, 40\].

4.1 Overview of Christian-Robinson Algorithm

In this work, the singular value decomposition (SVD)-based variant of the non-iterative OpNav algorithm, as described by Christian \[36\], is employed. The reader is directed to the literature \[12, 36, 40\] for an in-depth discussion of the brief summary that follows. Note that hereafter, the word *planet* is used to denote a planetary body, which also includes planetary moons, provided that they can be well-approximated as an ellipsoidal body.

4.1.1 Position Estimate from Limb Points

Let the camera frame \( F_C \) be defined with the \( z \)-axis aligned with the boresight of the camera, and the \( xy \)-plane lying on the image plane of the camera. Furthermore, consider a set of \( m \) detected limb points, given as pairs of pixel coordinates \( \{ u_i, v_i \}_{i=1}^m \). These pixel coordinate points can be related to a vec-
tor \( s_i \) in the image plane via

\[
s_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix},
\]

(5)

where \( K \) is the intrinsic camera calibration matrix [12,36,41]. Note that \( u_i, v_i \in \mathbb{R} \), since the limb points may be detected at sub-pixel level precision; this will be discussed in further detail in a subsequent subsection.

Then, the vector \( n \in \mathbb{R}^3 \) pointed from the spacecraft camera to the center of the body in the camera frame is obtained by solving the least-squares problem

\[
Hn = 1_{m \times 1},
\]

(6)

where

\[
H = \begin{bmatrix} s_1^T / \| s_1 \| \\ \vdots \\ s_m^T / \| s_m \| \end{bmatrix} = \begin{bmatrix} s_1^T \\ \vdots \\ s_m^T \end{bmatrix},
\]

(7)

and \( \bar{s}_i \) is the \( i \)th observed limb point on the image plane which relates to the coordinates in the image plane by

\[
\bar{s}_i = QT_P^C s_i,
\]

(8)

where

\[
Q = \text{diag}(1/a, 1/b, 1/c),
\]

(9)

with \( a, b, \) and \( c \) corresponding to the lengths of the planet’s principal axes, and \( T_P^C \in \mathbb{R}^{3 \times 3} \) is the rotation matrix from the camera frame to the planet’s principal axes frame, which is assumed to be known with known spacecraft attitude. In this work, \( T_P^C \) is assumed to be given by

\[
T_P^C = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix},
\]

(10)

where

\[
b_1 = r \times v / \| r \times v \|,
\]

\[
b_2 = -r / \| r \|,
\]

\[
b_3 = b_1 \times b_2.
\]

(11)

Note that the Moon specifically is a degenerate case where the commonly used ellipsoidal model is a sphere with \( a = b = c = R = 1737.4 \) km.

The position vector of the spacecraft’s camera in the camera frame, \( \hat{r}_C \), is obtained by

\[
\hat{r}_C = -(n^T n - 1)^{(1/2)} T_P^C Q^{-1} n,
\]

(12)

where \( T_C^P \in \mathbb{R}^{3 \times 3} \) is the rotation matrix from the planet’s principal axes frame to the camera frame. Finally, the position vector in the planet’s frame can be obtained via the simple transformation

\[
\hat{r}_P = T_C^P \hat{r}_C.
\]

(13)

4.1.2 Measurement Covariance Model

The measurement covariance associated with the Christian-Robinson algorithm using the Cholesky factorization is provided in [40]. Here, its adaptation to the SVD-based algorithm is provided.

The covariance of the least-squares problem solution \( n \) is given by

\[
P_n = \left( H^T \text{diag}(1/\sigma_{y_1}^2, \ldots, 1/\sigma_{y_m}^2) H \right)^{-1},
\]

(14)

where

\[
\sigma_{y_i} = J_i Q T_P^C R_x T_P^C Q^T J_i^T,
\]

(15)

\( R_x \) is the covariance of the horizon measurements, and

\[
J_i = \frac{1}{\| \bar{s}_i \|} n^T (I_{3 \times 3} - \bar{s}_i \bar{s}_i^T).
\]

(16)

As highlighted by Christian [40,42], \( R_x \) can be approximated by the relation

\[
R_x \approx \left( \frac{\sigma_{\text{pix}}}{d_x} \right)^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},
\]

(17)

where \( \sigma_{\text{pix}} \) is the standard deviation of the observed horizon points in units of pixels, and \( d_x \) is the pixel pitch in terms of radians, as defined in [12,40].

Then, using \( P_n \) from equation (14), the covariance of the position estimate in the camera frame is given by

\[
P_{rr} = FP_n F^T,
\]

(18)

where

\[
F = -(n^T n - 1)^{(1/2)} T_P^C Q^{-1} \left( I_{3 \times 3} - \frac{nn^T}{n^T n - 1} \right).
\]

(19)

The measurement covariance of the position in the planet’s frame is finally obtained via

\[
P_{rr} = T_C^P P_{rr} T_C^P.
\]

(20)

The value for \( \sigma_{\text{pix}} \) is a critical quantity that dictates the measurement covariance. While in the case of a real image, \( \sigma_{\text{pix}} \) can be estimated from the camera’s point spread function (PSF), this is not available for rendered images as is the case in this work. Instead, in this work, an approximate value for \( \sigma_{\text{pix}} \) is obtained by converting the limb point coordinates \( \{u_i, v_i\}_{i=1}^m \) to a polar representation \( \{\rho_i, \theta_i\}_{i=1}^m \) with respect to the image center \( (u_p, v_p) \). This process assumes the camera’s boresight is aligned with the center of the imaged body. Note that this assumption is merely used to obtain an approximate value for \( \sigma_{\text{pix}} \), but the position estimate \( \hat{r}_C \) from the OpNav...
algorithm is obtained solely through equation (12) and does not make use of this assumption. Once the polar coordinate representations of the limb points are obtained, $\sigma_{\text{pix}}$ is approximated as the standard deviation of the radii $\rho_i$.

4.2 Detection of Lit Limb

As described in the previous subsection, the horizon-based OpNav necessitates a set of limb points $\{u_i, v_i\}_{i=1}^m$. The detection of the lit limb consists of three principal steps, as outlined in [36]. The first step consists of detecting the limb at the pixel level using the Canny edge detection algorithm [43]. This is followed by filtering these detected points based on the illumination direction within the image plane, and finally a sub-pixel level refinement of the filtered pixel-level limb points.

For the first step, various edge detectors can be used, and a comparative assessment is provided in [36] for the interested reader. The second step consists of scanning the image along the direction of the sunlight projected onto the image, which is obtained by

$$u'_\text{illum} = \frac{u_{\text{illum}}}{\|u_{\text{illum}}\|},$$

(21)

where

$$u_{\text{illum}} = -\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} T^v \mathbf{d}_\odot.$$  

(22)

Note that $\mathbf{d}_\odot$ is known from planetary ephemerides alone, irrespective of knowledge of the spacecraft’s position. Along each scan, the first pixel with a pixel intensity higher than a pre-defined threshold $\epsilon$ is recorded as a “coarse” limb point. Based on these coarse limb points, an $n_u$-by-$n_v$ binary masking matrix $\mathbf{M}$ is constructed; $\mathbf{M}$ has 1’s on $n_M$-by-$n_M$ submatrices centered at the coarse limb point coordinates, and zeros elsewhere. Note $n_M < n_u$, $n_M < n_v$, and $n_M$ must be odd. In this work, a threshold of $\epsilon = 20$ and $n_M = 5$ is used. Once $\mathbf{M}$ is constructed, the Hadamard product of $\mathbf{M}$ and the Canny edge pixels in matrix form returns the filtered edge pixels near the coarse limb points. The final refinement is done using the Zernike moment-based technique described in [36]. This involves computing a corrective term on the filtered pixel-level edge coordinates, $\{\tilde{u}_i, \tilde{v}_i\}_{i=1}^m$, based on the Zernike moment information of a local, $N$-by-$N$ patch centered at $\{\tilde{u}_i, \tilde{v}_i\}_{i=1}^m$. The expression for the final sub-pixel level edge, $\{u_i, v_i\}_{i=1}^m$, is given by

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} \tilde{u}_i \\ \tilde{v}_i \end{bmatrix} + \frac{Nl}{2} \begin{bmatrix} \cos \psi \\ \sin \psi \end{bmatrix},$$

(23)

where

$$\psi = \arctan2(\text{Im}(A_{11}), \text{Re}(A_{11})), \quad (24)$$

and

$$l = \left[ 1 - w^2 - \sqrt{(w^2 - 1)^2 - 2w^2 A_{20}/A_{11}} \right] / w^2.$$  

(25)

Here, $w$ is the width parameter, set to 0.5 in this paper, and $A_{nm}$ are Zernike moments of the original image, whose expressions are given in [36].

Figure 7 shows the resulting limb points from the three steps. With the Canny edge detection alone, craters casting sharp shadows on the lunar surface are falsely identified, as shown by the red points. The illumination scans, shown by the orange lines, run from the top left to the bottom right along the sunlight direction and return the “coarse” limb points shown by the blue stars. Applying a 5-by-5 mask around each coarse limb point, the Zernike moment-based sub-pixel edge refinement is applied, to finally obtain the lime green points.

4.3 NRHO-Specific Considerations for OpNav

There are considerations for using OpNav specifically on NRHO that warrant attention. Firstly, the large variation in range, at around 3300 km around perilune and around 70,000 km around apolune for a 9:2 NRHO, poses a challenge in terms of choosing one (or multiple) appropriate camera(s). In addition, the orientation of the NRHO consisting of its apolune lobe extending below the Moon’s pole (in the case of the Southern L2 NRHO, extending below the South pole) results in illumination conditions with a periodic structure.
Fig. 7: Limb points detected on a rendered image of the Moon with terrain features. Crater limbs are identified by the Canny edge detector (shown in red) but are filtered out by applying a mask based on the coarse limb points (shown in blue). The lime green points correspond to sub-pixel level edges obtained based on the filtered, pixel-level edge from the Canny detector. The faded effect around the limb is due to the Gaussian blur that is applied to the raw rendered image.

4.3.1 Variation of Distance to the Moon

The range from the spacecraft camera to the observed body is a critical parameter that one must keep in mind when considering OpNav-based measurements. A critical issue for the horizon-based OpNav lies in the decreasing observability with range. To illustrate this, consider the apparent angular diameter of the Moon, \( \delta \), as a function of the range \( r \). Through simple geometry, the expression for \( \delta \) can be obtained as

\[
\delta = 2 \arctan \left( \frac{R}{\sqrt{r^2 - R^2}} \right), \tag{26}
\]

where \( R = 1737.4 \) km is the radius of the Moon. Figure 8 shows this relationship for a range between 3,000 km and 75,000 km. Due to the nature of the expression (26), the apparent angular diameter rapidly falls with increasing ranges under 10,000 km, then tapers off with a much slower decrease at ranges over 10,000 km. This results in two complications vis-à-vis OpNav: due to the variation of an “ideal” field of view (FOV) camera at different ranges from the Moon, and due to the fundamental reduction of the observability of the range at larger distances.

Since OpNav involves solving the least-squares problem from the limb pixel coordinates, it is preferable to have the Moon occupying a significant portion of the image. Near apolune of an NRHO, this would correspond to a small FOV of a few degrees; however, as the spacecraft approaches its perilune, an increasingly larger FOV is required for the Moon to fit within the image. Hence, while this is outside of the scope of this work, one may consider using multiple cameras to be operated at different range bands, or a variable FOV camera, in order to provide the ideal images to the OpNav algorithm.

The observability issue is more fundamental to the limb-based, range-finding problem; this is governed primarily by the mathematical expression (26) and cannot be circumvented. Rather, this pushes operation planning to prioritize collecting images for OpNav closer to the Moon, where the observability in the range is better, over collecting images further away.

4.3.2 Variation of Illumination Condition

The illumination condition of the Moon, when observed from an NRHO, varies between full Moons, half Moons, to new Moons, through a predictable pattern. Neglecting Earth eclipses, since the apolune of the NRHO is located at about 70,000 km below the south pole of the Moon, a half Moon is visible at any time around apolune. In contrast, when approaching perilune, the lighting condition on the Moon varies depending on the pass; if the Moon is at its closest or furthest point from the Sun along its orbit around the Earth, the camera sees about half of the Moon throughout its perilune pass. Meanwhile, if the Moon is precisely at the halfway point between the closest and furthest point from the Sun along its orbit,
the spacecraft sees a switch between a full Moon and a new Moon through its perilune pass.

The variation in illumination condition along the NRHO is quantified through the phase angle (Sun-Moon-spacecraft angle) \( \phi \), given by

\[
\phi = \arccos \left( \frac{\vec{r} \cdot \vec{d}}{|\vec{r}| |\vec{d}|} \right). \tag{27}
\]

This angle is shown against the z-coordinate along the NRHO in the Earth-Moon rotating frame in Figure 9 for a duration of approximately 4 periods. In this Figure, perilunes occur when \( z \) is at its largest value along each revolution, while the apolunes occur when \( z \) is at its most negative value along each revolution. As explained in the previous paragraph, \( \phi \) is centered around 90° around apolunes, while a periodic fluctuation trend is observed around perilunes; specifically, tracking the trace from the initial time at apolune, the initial perilune involves flying from the Sun-lit side (small \( \phi \)) to the dark side (large \( \phi \)), as shown by the yellow trace, while the next perilune flies over the terminator, resulting in \( \phi \approx 90° \) throughout, as shown by the light green trace. The perilune flying over the Sun-lit/dark side and over the terminator alternates, as expected.

4.4 Navigation Filter

The operation of a spacecraft requires a navigation filter to provide an estimate of the spacecraft’s state given a sequence of OpNav measurements. For this purpose, we implement an extended Kalman filter (EKF) \[44\]. This recursive filtering technique has seen great popularity and success throughout the history of spaceflight \[45\], primarily due to its simplicity and low computational burden. In this section, we follow the notation in \[17\] and let \( \theta = [r_1^T, \dot{w}_1^T]^T \) denote the spacecraft’s state, evolving by (1) as \( \dot{\theta} = f(t, \theta) \). The EKF is implemented in discrete time \( \{t_j \in \mathbb{R} : t_{j+1} > t_j, j \in \mathbb{N} \} \), and we write the (marginal) filtering posterior of the state vector \( \theta_a = \theta(t_a) \) given a sequence of measurements \( \{y_k\}_{k=0}^N \) by the probability density function \( p(\theta_a|\{y_k\}_{k=0}^N) = \mathcal{N}(\tilde{\theta}_{a|b}, \Sigma_{a|b}) \). The filter updates the estimates recursively from a Gaussian prior \( \mathcal{N}(\tilde{\theta}_{0|0}, \Sigma_{0|0}) \) in a time prediction followed by a measurement update, described separately below.

4.4.1 Time Prediction

To account for uncertainties in (1) we include additive process noise \[44\], resulting in the time update

\[
\dot{\theta}_{j|j-1} = \dot{\theta}_{j-1|j-1} + \int_{t_{j-1}}^{t_j} f(\tau, \dot{\theta}_{j-1}(\tau))d\tau, \tag{28}
\]

\[
\Sigma_{j|j-1} = \Phi(t_j, t_{j-1}) \Sigma_{j-1|j-1} \Phi(t_j, t_{j-1})^T + Q, \tag{29}
\]

where (28) is evaluated using an explicit embedded Runge-Kutta Prince-Dormand (8,9) method from the GNU scientific library \[46\], and \( \Phi(t_j, t_{j-1}) \) is the state-transition matrix obtained by numerically integrating the initial value problem

\[
\begin{align*}
\Phi(t_j, t_{j-1}) &= \left( \frac{\partial f(t, \theta)}{\partial \theta} \right)_{(\tau, \theta_{j-1}(\tau))} \Phi(t, t_{j-1}), \\
\Phi(t_{j-1}, t_{j-1}) &= I_{6 \times 6}.
\end{align*}
\tag{30}
\]

The process noise is modeled as an unbiased random process, and its covariance \( Q \) is taken to be a function of the sample period \( h_j = t_j - t_{j-1} \), as

\[
Q = \sigma_u^2 \begin{bmatrix}
\frac{1}{3} h_j^3 & \frac{1}{2} h_j^2 \\
\frac{1}{2} h_j^2 & h_j
\end{bmatrix}, \tag{31}
\]

where \( \sigma_u^2 \) is the diffusion coefficient to be adjusted to optimize the filter performance \[47\].

4.4.2 Measurement Update

The OpNav measurements are modeled as direct position measurements with additive Gaussian noise

\[
y_j = E \theta_j + r_j, \quad r_j \sim \mathcal{N}(0, R_j), \quad E = \begin{bmatrix} I & 0 \end{bmatrix}, \tag{32}
\]

Fig. 9: z-position in the Earth-Moon rotating frame against phase angle \( \phi \) along an NRHO over a duration of approximately 4 periods. The red cross indicates the z-position and phase angle at the initial time.
resulting in the linear standard measurement update

\[ K_j = \Sigma_{jj}^{-1}E^T \left( E \Sigma_{jj-1}E^T + R_j \right)^{-1}, \]  

\[ \hat{\theta}_{jj} = \hat{\theta}_{jj-1} + K_j \left( y_j - E \hat{\theta}_{jj-1} \right), \]  

\[ \Sigma_{jj} = (I - K_jE) \Sigma_{jj-1}. \]

Both the measurement \( y_j \) and the measurement noise covariance \( R_j \) are direct outputs of the OpNav algorithm, in \([13]\) and \([20]\) respectively.

5. Experiments

The OpNav simulation framework is used to generate images along an NRHO and run the Christian-Robinson algorithm to construct position measurements with associated measurement covariance. The measurements are then incorporated into an EKF.

The camera is considered to have an image size of 2048-by-2048 pixels, with a focal length \( f = 240 \) mm and a sensor size \( s = 100 \) mm along each direction, which corresponds to FOV of about 23.54°. As shown by Figure 3, this camera can thus image the full Moon at ranges above \( \approx 8500 \) km. To account for operational considerations on pointing the camera, images are generated at ranges equal to or above 10,000 km. The detector array skewness in the camera frame’s \( z \)-direction (boresight). Meanwhile, the effect of this pixel shift as follows: since the image contains 2048 pixels in each direction, the physical distance represented within a single pixel at a range of 70,000 km can be computed from the focal length \( f \) and the sensor size \( s \) as \( 100 \times 70000/240 \approx 14 \) km. If the Moon is shifted by half of a pixel from its true position, we should expect a bias of about 7 km at a range of 70,000 km, which coincides with what we can observe in Figure 11c.

5.2 Effect of Terrain

The effect of the lunar terrain as displacement on the ellipsoidal mesh is observed in Figure 11, where the measurements are generated from images with terrain, without anti-aliasing, and with half-pixel compensation. This may be compared against Figure 10d which uses the same settings apart from omitting the terrain. Note that, unlike the half-pixel shift, the terrain alters the dimensions and orientation of the best-fitting ellipsoid from the model that is given to the Christian-Robinson algorithm via equation (9). Recall that in this work, a spherical Moon model based on values from PCK is used; modifying the triaxial model’s dimensions and orientation based on recent lunar geodetic data may reduce the error due to terrain. The error on \( \hat{r}_C \) appears on all three components. Looking at the last row in Table 2 it is possible to see that both the mean and standard deviation of the measurement errors worsen with terrain in all components.

5.3 Comparison of Observability per Camera-Frame Component

Across all cases in Figure 10 and 11, it is possible to observe that the error in the camera \( z \)-direction is
about two orders of magnitude worse than in the $x$- and $y$-directions. This is a manifestation of the difference in observability along the camera boresight compared to the directions on the plane perpendicular to the boresight. Since the apparent diameter of the Moon varies slowly as the range increases, the observability in this direction is poorer than in the other two directions.

Also, in all three components, a smaller range corresponds to the Moon occupying a bigger portion of the image, generally resulting in a greater number of limb points $m$. Particularly for the (Southern) NRHO, since at ranges beyond 10,000 km the spacecraft is sufficiently below the Earth-Moon rotating plane, the illumination condition has a relatively minor impact on $m$. Therefore, the error of $\hat{r}_C$ is smaller at shorter ranges, and increases with range.

5.4 Measurement Covariance

Figure 12 shows the error in range (taken as the difference between the estimated and true range), as well as the error by camera-frame component, against epoch. The images used for this Figure correspond to those from Figure 11. For the error by component,
Table 2: Statistics of measurement error processed through Christian-Robinson algorithm

<table>
<thead>
<tr>
<th>Fig.</th>
<th>Settings</th>
<th>Mean error, km</th>
<th>Standard deviation, km</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Anti-alias</td>
<td>Half-pixel comp.</td>
<td>Terrain</td>
</tr>
<tr>
<td>10a</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>10b</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>10c</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>11</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

Fig. 11: Effect of terrain on position measurement error in the camera frame, shown against range from the Moon. Images are rendered with terrain, without anti-aliasing, and with half-pixel compensation.

The red shaded region corresponds to 3-σ values associated with the measurement, computed by equation (18), which seems to successfully capture the spread of the measurement accurately. The periodic structure in the measurement error as well as the 3-σ of each component is a manifestation of the NRHO’s (quasi)-periodic motion, where measurements near perilunes have lower error (and correspondingly lower 3-σ’s) while measurements near apolunes are worse.

5.5 Navigation Filter Performance

The horizon-based OpNav measurements are incorporated into an EKF to provide state estimates of a spacecraft on the NRHO. An initial filter prior $\Sigma_{0|0}$ is given by

$$\Sigma_{0|0} = \text{blockdiag}(\sigma_{r,0}^2 I_{3\times3}, \sigma_{v,0}^2 I_{3\times3}).$$

Fig. 12: Position measurement error in range and camera-frame component shown against epoch. The red shaded region corresponds to measurement 3-σ’s computed from equation (18).

where $\sigma_{r,0} = 10$ km, and $\sigma_{v,0} = 10$ cm/s. The diffusion coefficient in the process noise is given by $\sigma_u = 10^{-7}$ m/s$^{3/2}$. The dynamics of the filter are varied compared to the true dynamics of the spacecraft by increasing the SRP ballistic coefficient $m/A$ by 50%. The initial state estimate $\hat{\theta}_{0|0}$ is given by

$$\hat{\theta}_{0|0} \sim \mathcal{N}(\theta(t_0), \Sigma_{0|0}).$$
Figure 13 shows the results of the EKF in terms of state estimate error and standard deviations, in components of the filter’s frame. Overall, the filter is able to reduce the initial filter covariance in both position and velocity and shows a consistent performance where the state estimate lies within the filter 3-$\sigma$ over the vast majority of time. The periodicity of the NRHO motion is visible yet again; this is a combined effect of the periodic nature of the quality of the measurements, as seen in Figure 12, as well as the dynamics itself, where the spacecraft state evolves rapidly around perilune and slowly around apolune.

The spikes observed in the velocity estimates correspond to perilune passes, where the spacecraft rapidly changes its direction of travel as it flies around the Moon. This results in even a small state estimate errors prior to perilune to cause significant error in the velocity component-wise. Nevertheless, this effect is quickly attenuated by measurement updates, as the velocity estimate error quickly reduces immediately after the perilune passes.

Finally, we note that accurate state knowledge near apolune is critical for autonomous stationkeeping, as control maneuvers are commonly executed near apolune, where the dynamics is less sensitive to navigation and maneuvering errors. The filter provides apolune position estimates with errors under ~5 km and velocity estimates with errors consistently below 2.5 cm/s, which lie between the low (1 km, 1 cm/s) to medium (10 km, 10 cm/s) navigation error magnitudes studied in [48]. The OpNav navigation performance suggests the ability to conduct OpNav-only autonomous GNC on NRHOs.

6. Conclusions

In this work, a high-fidelity simulation environment for horizon-based OpNav has been implemented. The process for choosing the appropriate software, along with the steps involved in configuring the environment, has been discussed. The simulator includes the imaged celestial body including surface features, a directional light source with an angular diameter, and any additional eclipsing body. While this work has focused on using the simulation environment along an NRHO, the same environment could be used for any cislunar orbit without any modification, and for any other horizon-based OpNav applications relative to other celestial bodies by merely swapping the primary body as well as any eclipsing bodies. Images generated through an NRHO have been processed through the Christian-Robinson algorithm and have been incorporated into an EKF, which demonstrated satisfactory navigational performance.

The effect of anti-aliasing as well as flooring/ceiling done by rendering engines on horizon-based OpNav has been studied. Furthermore, in the context of using this framework on an NRHO, the impact of including the lunar terrain, as well as the periodic structure on the illumination, and challenges due to the large variation in range, has been discussed. The latter characteristic of the NRHO poses a challenge for OpNav, the nature of horizon-based OpNav relying on the apparent diameter poses a fundamental degradation of observability along the camera’s boresight as the range increases.

Numerical experiments using rendered images are conducted with state-of-the-art methods for identifying the lit limb, which is then converted to position measurements via the SVD-based variant of the Christian-Robinson algorithm. Impacts of rendering effects, the lunar terrain, as well as the large variation in range along NRHOs on the quality of the position measurements, are studied. The imperfection of the obtained measurements highlights the limitation of using analytical models of the lit limbs, with assumptions on things such as zero-mean Gaussian error on identified limb pixels, or on consistent Sun phase angle.

References


Fig. 13: EKF results using horizon-based OpNav measurements along the NRHO from Figure 11. The position (left column) and velocity (middle/right columns) errors are components in the J2000 frame, which is the frame used within the filter. The right column corresponds to the same data as the middle column, zoomed in.


