JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 58, No. 4, July–August 2021

Centroid and Apparent Diameter Optical Navigation on Mars Orbit

Thibaud Teil* and Hanspeter Schaub†
University of Colorado, Boulder, Colorado 80309-0431
and
Daniel Kubitschek‡
University of Colorado, Boulder, Colorado 80303-7814

https://doi.org/10.2514/1.A34815

The work described in this paper harnesses an open-source astrodynamics engine and visualization to quantify the performance of on-board optical navigation. It introduces the Hough Circles transform as a candidate method for centroid and apparent diameter extraction. The coupled nature of the simulation enables simultaneous pointing and orbit determination with dynamic image generation, all in a realistic flight-software environment. Navigation is done about Mars solely using optical images, and by means of limb or centroid/diameter extraction. Through the implementation of pre-existing algorithms for a baseline comparison and the development of Hough Circles, an end-to-end autonomous flight-software stack is developed and tested. This research provides insight into achievable navigation accuracy and image processing methods, as well as outlier mitigation for mission readiness.

I. Introduction

OPTICAL navigation (OpNav) in astrodynamics refers to the use of images taken by an on-board camera in order to determine the spacecraft's position [1]. The images contain solar system bodies and therefore provide relative position and attitude information. Commonly, OpNav measurements are combined with radiometric data or other measurements to compute a navigation solution. Nonetheless, the images can provide all the necessary information to estimate the spacecraft states, which makes OpNav a good candidate data type for autonomous navigation.

Deep space missions (e.g., Deep Space 1 [2], Stardust, or Deep Impact [3]) rely heavily on OpNav. Nevertheless, only Deep Space 1 and Deep Impact used autonomous OpNav (AutoNav [2]). Both were successful and built confidence in the potential use-cases for autonomous navigation. Optical images are also used for entry, descent, and landing (EDL), as seen in practice with the DIMES [4] system used to land Mars rovers. These use cross-correlation methods [5,6] developed specifically for EDL.

Yet, autonomy is also valuable during routine mission operations, and presents a different kind of challenge than during mission-critical phases. Both scenarios require high levels of fidelity and robustness to a wide set of conditions, but with more frequent use of the algorithms comes more exposure to potential faults. One reason that confidence in the on-board algorithms is difficult to achieve lies in the difficulty in reproducing flight-like conditions on Earth. A natural remedy is to provide a general simulation framework that allows to test algorithms in a wide set of conditions.

This paper combines several contributions in order to deliver a full mission analysis tool. In addition, this work develops open-source code bases allowing for continuous validation, testing, and community support: the module documentation, and all scenarios run in this paper are available for the reader (http://hanspeterschaub.info/basilisk). The scenarios developed in this paper focus on the autonomous navigation about Mars, but the ease in scriptability easily extends the work to other planetary bodies because there is little in the methods that depend on the central body. The novel results presented can be summarized in the following points:

1) The simulation is an extension to capabilities such as AutoNav [2]: it provides an end-to-end flight software (FSW) stack, coupled with a visualization, and dynamics engine that embed all of the necessary components for simulating various OpNav scenarios. The addition of more advanced image processing methods instead of center-of-brightness, and application to on-orbit navigation simulation are new results in this complete framework.

2) For its image processing component, this work implements a state-of-the-art limb-based OpNav method, while introducing a robust Hough Circle [7] finding algorithm. Although the circle finding algorithm has been used extensively in robotics, its applications to the spacecraft navigation field have primarily been for crater detection [8,9] and has not been applied to center and apparent diameter (CAD) navigation.

3) Building off previous work in Ref. [10] exploring closed-loop optical navigation simulations, this paper develops novel results that show the robust performance of simultaneous orbit determination (OD) and pointing autonomous OpNav scenarios. Previous work by the authors had only used the images for pointing, whereas this paper also performs orbit determination. Furthermore, results are evaluated through Monte Carlo simulation, and in the presence of faulty images to display the robustness of the FSW stack and Hough Circles algorithm.

This research presents simultaneous pointing and OD OpNav on orbit using circle fitting instead of ellipse fitting, as well as comparative capabilities for different methods in an ideal flight-like simulated environment. Given the pairing of the pointing tasks that centers the planet on the image (hence limiting the camera projection errors) and the use of planets that are not largely oblate (e.g., telluric planets and the moons of gas giants), the results show that circles can provide comparatively good navigation solutions. Indeed, the results of Hough Circles are compared with an ellipse fitting method in order to display the capabilities of circle fitting for coarse navigation. The state-of-the-art algorithm implemented as a baseline comparative algorithm is the noniterative horizon-based method by single value decomposition (SVD) [11]. The comparison to SVD is illustrative, in order to show that a geometrically exact method does not provide vastly different results. With this in mind, the Canny transform is the limiting factor for both the methods. The later analysis exemplifies the tool's flexibility and provides comparative results to illustrating how circle fitting can, in some cases, provide a robust image processing method.

The performance of the SVD method, paired with customized pixel-level edge detection, then enhanced with subpixel refinement (Zernike Moments [12]) provides the highest levels of performance for OpNav on full disk images. Further studies have shown that edge

Received 17 March 2020; revision received 19 November 2020; accepted for publication 25 November 2020; published online 12 March 2021.

*Graduate Research Assistant, Colorado Center for Astrodynamics Research, Department of Aerospace Engineering Sciences, 431 UCB.

†Glenn L. Murphy Chair of Engineering, Department of Aerospace Engineering Sciences. Fellow AIAA.

‡Flight Dynamics Lead, Laboratory for Atmospheric and Space Physics, 1234 Innovation Drive.

localization near the luminance poles were unreliable, and optimized the limb fit to use at most 140° of the lit arc [13]. These developments have found to improve overall performance, allowing centroid localization being certain within 0.2 pixel 1-σ bounds, but require additional processes to optimize the results. Furthermore, iterative methods like random sample consensus (RANSAC) are implemented in order to reject outliers and provide robust feature detection. Part of the motivation for analyzing Hough Circles for OpNav is to provide a robust, quick-to-implement, reliable method, in a single algorithm. Hence, the ellipse-based method SVD is implemented with the Canny transform [14] (edge detection algorithm) in order to directly compare OpNav solutions knowing that improvements for both methods would enhance results.

The proposed results are useful to design and develop flight algorithms for specific mission requirements (e.g., provide spacecraft ephemeris knowledge within 10 km) as well as comparative capabilities for different methods in a flight-like simulated environment. This is useful to develop technology tests in determining what level of complexity needs to be implemented on board for a specific requirement to be satisfied, depending on the camera quality and required accuracy and robustness. All scenarios run in this paper can be found in the src/examples/OpNavScenarios/ folder within Basilisk.

II. Simulating a Mars OpNav Orbiter

The scenario referenced in this research places a spacecraft on Mars orbit using only CAD for OpNav. This scenario (pictured in Fig. 1) enables testing of a full OpNav sequence, from taking images to control. This section develops the simulation used for this example scenario. This architecture harnesses two main components: a high-fidelity, faster than real-time, astrodynamics simulation framework Basilisk [15,16], and a sister software package (Vizard [17]) to dynamically visualize the simulation environment [10]. Both are developed by the Autonomous Vehicle Systems (AVS) Lab and the Laboratory of Atmospheric Space Physics (LASP).

Basilisk (http://hanspeterschaub.info/basilisk) is a highly modular astrodynamics simulation framework that allows for the rapid simulation of complex spacecraft dynamics. Key features include solar radiation pressure [18–20], imbalanced reaction wheels [21], flexible solar panels [22], fuel slosh [23,24], as well as multiple body gravity and gravitational spherical harmonics. An associated visualization is built using the Unity gaming engine and is called Vizard. Here the Basilisk simulation messages are streamed directly to the visualization to illustrate the spacecraft simulation and environment states. At a high level, Basilisk performs all of the astrodynamics simulation duties, whereas Vizard generates synthetic images based on the simulation requests. More information on the simulation architecture used here can be found in Ref. [10].

Figure 2 shows the OpNav FSW algorithms used in order to navigate autonomously off of an image. Many OpNav methods exist in which different features can be extracted. In general, the images are synthesized in Vizard and transmitted to the Basilisk FSW modules. The first module extract features (CAD or Limb points in this case)

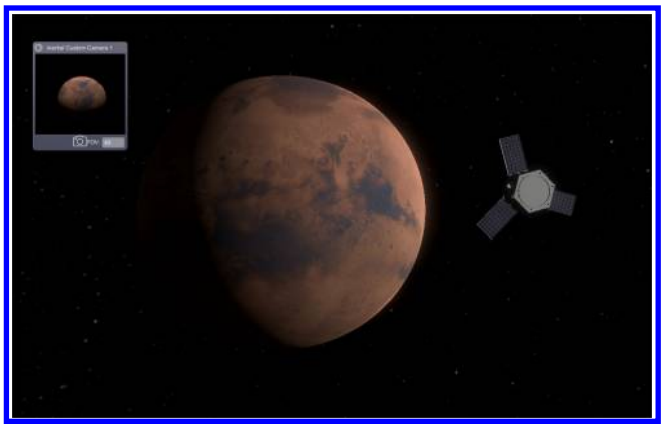


Fig. 1 OpNav on Mars orbit.

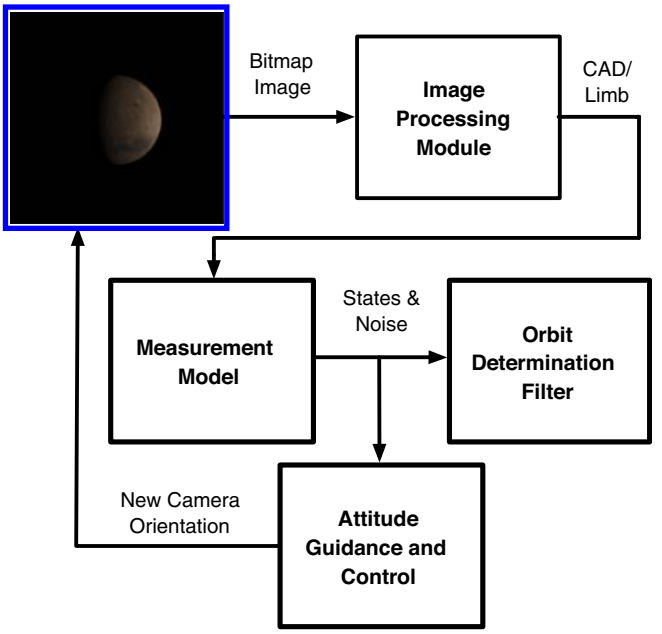


Fig. 2 Information flow between flight software modules. Each block represents a modular code base part of the guidance navigation and control algorithm stack.

and transmits them to a measurement model in order to generate a measurement and its covariance (measured state and noise). These states are used for attitude guidance and orbit determination.

Point distribution methods [25], star-occultation methods [26], and crater-tracking [27] are some of many feature extraction methods that provide promising results. However, they come at a computation cost. At the extreme, the current state-of-the-art for general OpNav is stereo-photoclinometry [28–30] (SPC), which allows the spacecraft to map and navigate the spacecraft environment with high precision. Nonetheless, it relies very heavily on Earth contact for its intensive image processing algorithms.

Although there is potential for using more computationally intense methods on-board, this research focuses on implementing CAD for autonomous OpNav. It provides the necessary information for on-orbit navigation: if the celestial object is resolved but not taking up the entire field of view. Figure 2 pictures the coupled nature of the simulation: the images processed are used for attitude guidance and control, which then generate new images. Star-tracker inertial attitude estimates, and camera frame knowledge (relative to the spacecraft body) are assumed to be available to the modules in this stack.

A. Synthetic Images

The images are rendered in a Unity camera simulator in realistic lighting conditions using either its integrated GPU ray-tracing capability or directional lighting. For the applications in this research, ray tracing does not change the images or results, and is disabled. Camera specifications, such as image size, frame rate, and focal length, are generated in the Basilisk camera module. The simulation provides corruption capabilities through the same camera module in order to render faulty images, although blooming and distortions can be natively added in the Unity visualization as well. Sensor effects are kept minimal for the initial study, and more corruptions are added in the last section. Future work could include perturbations from detector saturation, blooming, surface feature albedo, and more, if the mission application warrants that level of accuracy. Vizard is constantly improving as and additions can be tracked on the Basilisk website. For this application, Mars is modeled in the visualization as an oblate sphere with polar radius $R_{m.p} = 3376.2$ km and equatorial radius $R_{m.e} = 3396.2$ km.

The images are generated in a closed-loop manner: the spacecraft position and attitude determine the image generation. Furthermore attitude and trajectory control update the images live. This needs to be done in a fast and accurate manner. The difficulty comes in providing speeds for several-orbit simulations that tightly couple in attitude

Table 1 Camera parameters

$\boldsymbol{\sigma}_{\mathcal{CB}}$	${}^{\mathcal{B}}\boldsymbol{r}_{\mathcal{CB}}$ [m]	Image size (pixels)	Sensor size (mm)	FoV (°)	Frame rate (s)	International Organization for Standardization (ISO)
$[0\ 0\ 0]^T$	$[0\ 0.2\ 0.2]^T$	$[512\ 512]^T$	$[10\ 10]^T$	$[40\ 40]^T$	0.005	200

variations for image generation. The simulation runs with a 0.5 s time step, and takes 512×512 images every minute for attitude control; it simulates the required dynamics models, alongisde flight software algorithms, and the OpNav OD algorithms for a 10-hour-long orbit in approximately 1 min ($\sim$600 $\times$ real-time).

The camera parameters are given in Table 1 and use a square image with a wide field of view (FoV). The sensor size of 1 cm is equivalent to choosing a focal length of 1.373 cm given the relation

$$\tan\left(\frac{\text{FoV}}{2}\right) = \frac{\text{SensorSize}}{2f} \tag{1}$$

The position and orientation of the camera are arbitrary in this scenario, as long as they do not create any self-shadowing. This is avoided easily with the parameters implemented.

To better understand the limitation of circle fitting on an ellipsoid, a brief calculation is provided: the ratio of Mars's equatorial radius to its polar radius is $(R_{m,e}/R_{m,p}) = (3396.2/3376.2) = 1.0059$. When looking directly at Mars from $Z = 20{,}000$ km, the difference between accounting for this oblateness is conservatively $\Delta = (f/Z)(R_{m,e} - R_{m,p}) = 1.4 \cdot 10^{-5}$ m on the image plane. For the given camera parameters and at the worst case, this corresponds to a subpixel error (the pixels are $2.0 \cdot 10^{-5}$ m). The assumption is to neglect oblateness, and verify this hypothesis with the navigation results.

B. Simulated Astrodynamics

This section specifies the baseline simulation parameters as well as the FSW algorithms used. The total time simulated is 1000 min, allowing to go through a wide variety of images. The baseline orbit parameters were chosen in order to vary the OpNav conditions (lighting and apparent disk size) while staying out of the eclipsed region of the orbit. This choice is made to avoid losing lock on the planet, as the focus of the paper is not to study this part of an orbit phase.

Table 2 Spacecraft initial states

$\boldsymbol{\sigma}_{\mathcal{BN}}$	$\boldsymbol{\omega}_{\mathcal{BN}}$ (rad/s)	Orbital elements (a, e, i, Ω, ω, f)
$[0\ 0\ 0]^T$	$[0\ 0\ 0]^T$	(20,000 km, 0.6, 10° 25°, 190°, 90°)

Throughout this paper, the camera frame is noted $\mathcal{C}$ and is defined as having a Z axis along the camera boresight, an X axis positive along the camera columns, and a Y axis positive along the camera rows. The spacecraft body frame is $\mathcal{B}$, whereas the inertial frame is $\mathcal{N}$. Direction cosine matrices (DCMs) are noted $[\mathcal{BN}]$ to represent the rotation from the inertial frame to the body frame, which can also be represented as modified Rodrigues parameters (MRPs) $\boldsymbol{\sigma}_{\mathcal{BN}}$. The rotation rate of the body frame relative to the inertial frame is noted $\boldsymbol{\omega}_{\mathcal{BN}}$, and left superscripts denote the frame in which a vector is expressed in [31]. Positions are noted as $\boldsymbol{r}$, subscript $\mathcal{CB}$ represents the camera position relative to the spacecraft center of the frame B, and, similarly, $\boldsymbol{r}_{\mathcal{BN}}$ is the vector from the center of $\mathcal{N}$ to the spacecraft body frame.

The simulation uses SPICE (naif.jpl.nasa.gov/naif/) data, where the simulation begins on December 12th at 22:00 2019 (GMT). Given the initial conditions of the spacecraft in orbit (seen in Table 2) Mars first appears as a waxing crescent, and as the spacecraft reaches apoapse, Mars becomes full. Near the end of the simulation Mars begins to go through a waning crescent phase. Figure 3 shows images used for OpNav during the run as well as a plot of the phase angle and planet apparent diameter. This allows the FSW algorithms to be tested along a wide variety of lighting conditions and planet sizes though the results generalize to many different orbital parameters. To illustrate this, a Monte Carlo analysis is performed after the study of the baseline scenario.

The simulation modules assigned to modeling the spacecraft dynamics and environment are described in Table 3. These modules simulate spacecraft attitude gyroscopics and gravity [32], eclipse, reaction wheels [21], and star-trackers. Eclipsing is also modeled: this usually creates a halt in the spacecraft measurements. In a fully coupled pointing–OD simulation this means that the spacecraft enters a search mode, which intends to allow for the modeling of a fully independent spacecraft. In this scenario, a rough pointing to Mars can be accomplished even with inaccurate knowledge of the spacecraft position. Therefore, in order to focus on the OD solution, the spacecraft does not go through eclipse and undergoes a 5-min guided pointing mode (in which the planet location relative to the spacecraft is known) before

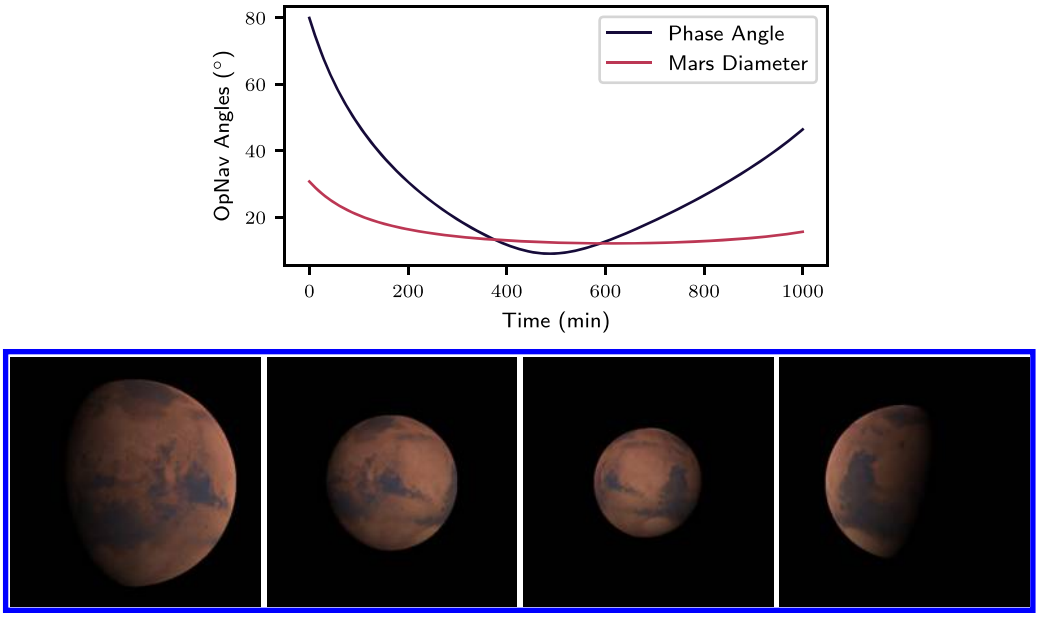


Fig. 3 Visuals of the scenario used for OpNav analysis. Important angles describing images (top) and sample images used for image processing (bottom).

Table 3 Simulation parameters: described in Basilisk documentation

Simulation modules instantiated	Necessary parameters at initialization
Spacecraft hub	Inertia $[I] = \text{diag}(900, 800, 600)$ kg $\cdot$ m^2, mass $M = 750$ kg
Gravity effector/eclipse	December 12th 2019 at 18:00:00.0 (Z), $\mu_{\text{mars}} = 4.28284 \cdot 10^4$ km^3/s^2, $\mu_{\text{earth}} = 0.3986 \cdot 10^6$ km^3/s^2, $\mu_{\text{jupiter}} = 1.26686 \cdot 10^8$ km^3/s^2, $\mu_{\text{sun}} = 0.327124 \cdot 10^{11}$ km^3/s^2
Simple navigation star-tracker	Attitude error $\epsilon_{\text{att}} = 1/3600°$, rate error $\epsilon_{\text{rate}} = 5 \cdot 10^{-5°}$/s
Reaction wheel effector	4 Honeywell HR16 wheels
Wheel orientations	Elevation angles (el): 40°, azimuths angles (az): 45°, 135°, 225°, 315° Positions in $\mathcal{B}$ [m]: $[0.8, 0.8, 1.79070]^T$ $[0.8, -0.8, 1.79070]^T$ $[-0.8, -0.8, 1.79070]^T$ $[-0.8, 0.8, 1.79070]^T$

starting to navigate and point using OpNav. Aside from these first 5 min (which do not appear on the plots shown in this paper) the spacecraft has no knowledge of its position relative to the planet.

Table 3 defines the spin axes of the wheels [31] in the body frame through the elevation and azimuth angles with the equation

$$^{\mathcal{B}}\hat{\boldsymbol{g}}_s = [\cos(el)\cos(az) \quad \cos(el)\sin(az) \quad \sin(el)]^T \tag{2}$$

More specific information on the wheel specifications is available on Honeywell documents (aerospace.honeywell.com), and all implementation details and variable definitions are fully described in Basilisk documentation. These remain key parameters to implement the scenario in full detail.

Beyond the astrodynamics modules, the simulation also implements several FSW algorithms. These are all developed in C for speed, and compatibility with heritage FSW. These are broken up in two groups: imaging FSW (summarized in Table 4) and pointing/OD FSW (Tables 4 and 5). The imaging modules encompass the OpNav raw measurements (limbs and circles) as well as the measurement models to provide spacecraft position. Some image processing methods are pulled from the OpenCV (docs.opencv.org/) library and more information is found in their code documentation. The Hough Circles module uses two main parameters: the first drives the sensitivity of the edge detection (canny_thresh) and the second defines the minimum number of votes required to be considered a circle (voteThresh). Finally the minimum (and potentially maximum) circle radius can also be set. These parameters are chosen in accordance with values used for general circle fitting from previous work [10] and have shown to find limbs in a wide variety of situations.

Table 5 shows the modules implemented for centroid-based pointing guidance [10] (associated Basilisk module is opNavPoint [https://tinyurl.com/y6f3kf8d]), OD, and control using MRP-Feedback [31]. The attitude feedback control is asymptotically stabilizing using the MRP coordinates, and the associated Basilisk module is MRP_Feedback (https://tinyurl.com/y4d86emd). The OD algorithm is in the module named "relativeODuKF" (https://tinyurl.com/y3y8jffj). The standard deviation of the error in pixel measurements is given by σ_{pix}.

III. Image Processing and Filtering

This section describes the details of the OpNav data chain from captured image to orbit estimate. This is done more modularly by taking the measurement model outside of the filter in order to more easily interchange models. The flow of data through the simulation is shown at a high level in Fig. 2, and is summarized in Table 4.

A. Limb Detection

In recent years, high-fidelity image processing algorithms have been developed in order to support autonomous navigation around the moon [33]. Given the desired application to spherical or ellipsoidal bodies, these algorithms generally extract an ellipse centroid [34], or fit a limb [35]. These methods provide accurate measurements of a spacecraft's relative position to the body provided an unambiguous limb.

As stated in the Introduction, the reference method chosen to measure spacecraft position is the noniterative horizon-based optical navigation by singular value decomposition [11,36] (SVD). This method was chosen for its high performance not only analytically (geometrically exact if a clear limb is provided) but also numerically. The method takes a set of limb points as an input and outputs a camera position in the planet frame. The algorithm is briefly summarized here.

Assume that a set of N limb point vectors $\boldsymbol{s}_k = (x_k, y_k, 1)$ (km) are given in the camera frame and in the image plane (normalized focal length). Pairs (x_k, y_k) are the position of the kth limb point in the

Table 4 Flight software for imaging: described in Basilisk documentation

Flight software modules instantiated	Necessary parameters at initialization
Image processing (arguments for Hough Circle method)	canny_thresh = 100, minDist = 50 pix minRadius = 20 pix, dp = 1, voteThresh = 30
Limb finding (arguments for Canny transform)	cannyThreshLow = 50, cannyThreshHigh = 100 blurSize = 3
Pixel line transform	Planet target is Mars, $\sigma_{\text{pix}} = 5.5$
Horizon Nav	Planet target is Mars, $\sigma_{\text{pix}} = 5.5$

Table 5 Flight software pointing and orbit determination

Flight software modules instantiated	Necessary parameters at initialization
OpNav point	minAngle = 0.001°, timeOut = 100 s $\omega_{\text{search}} = [0.06, 0.0, -0.06]°$/s, $Ch_c = [0, 0, 1]$ m
relativeOD	$\alpha = 0.02$, $\beta = 2$, $\kappa = 0$ $\boldsymbol{r}_{\text{error}} = [10, 10, -10]$ km, $\boldsymbol{r}_{\text{error}} = [0.1, -0.01, 0.01]$ km/s
MRP feedback RW	$K = 3.5$, $P = 30$ (no integral feedback)
RW motor torque	Control axes are $^{\mathcal{B}}[\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3]$

camera frame. Because of the FoV of the camera, a correction is added to radially centralize the pixels to compensate for magnifying effects. This is done using a simple model seen in the literature, and in accordance with Unity's camera models: an equidistant projection function [37]. Each detected point of the limb (x_k, y_k) is described in polar coordinates: $l_d = \sqrt{x_k^2 + y_k^2}$ and $\phi = \arctan 2(y_k/x_k)$, and l_d represents the distorted distance from the image center (or principal point). For $f = 1$, in the image plane, the model simply predicts that the true distance from the center is $l_u = \arctan(l_d)$. Therefore, each point's distance from the center is scaled by $l_d/\tan(l_d)$ before moving back to Cartesian coordinates.

$$[B] = [Q_m][\mathcal{PC}] \tag{3}$$

$$\bar{s}_k = [B]s_k \quad \text{and} \quad \bar{s}'_k = \frac{\bar{s}_k}{\|\bar{s}_k\|} \tag{4}$$

The planet-fixed frame is given by

$$[Q_m] = \text{diag}\left(\frac{1}{r_a}, \frac{1}{r_b}, \frac{1}{r_c}\right)$$

where r_a, r_b, r_c are the radii of the potentially ellipsoidal body along its principal axes. $[B] = [Q_m][\mathcal{PC}]$, where $\mathcal{P}$ is the planet frame. In this paper the planet frame is taken as the inertial frame $\mathcal{N}$, and $[Q_m]$ represents Mars as flattened at the poles with $r_a = r_b = R_{m,e} = 3396.2$ km and $r_c = R_{m,p} = 3376.2$ km. After rotation and normalization according to Eq. (5), these are concatenated in order to solve for the cone that links the limb points to the focal point (characterized by the vector $\boldsymbol{n}$):

$$[H]\boldsymbol{n} = \mathbf{1}_{N\times 1} \quad \text{where} \quad [H] = [\bar{s}'_0 \ \ \dots \ \ \bar{s}'_N]^T \tag{5}$$

$${}^{\mathcal{C}}\hat{\boldsymbol{r}}_{\mathcal{BN}} = -(\boldsymbol{n}^T\boldsymbol{n} - 1)^{-(1/2)}[B]^{-1}\boldsymbol{n} \tag{6}$$

This is done by performing a QR decomposition on $[H]$, which constructs $[Q_H]$ orthonormal and $[R_H]$ upper triangular such that $[H] = [Q_H][R_H]$. This leads to the equation $[R_H]\boldsymbol{n} = [Q_H]^T\mathbf{1}_{N\times 1}$, which is solved by backsubstitution. Once $\boldsymbol{n}$ is computed, the spacecraft position is given in the camera frame by Eq. (6).

The only computation remaining rotates the position vector into the desired frames. This is done using star-tracker estimates of $[\mathcal{BN}]$ alongside the known camera frame $[\mathcal{CB}]$, which allows the module to output the covariance and estimates in the body, inertial, and camera frames for downstream use. The covariance manipulation from the measurement uncertainty is done according to Refs. [36,38].

This method is implemented in C (under the Basilisk module name of "horizonOpNav" [https://tinyurl.com/y5qqvuqx]). The last thing to do in order to implement this method fully is to create a limb finding method. This is done by using Canny transform [14] implemented in OpenCV, preceded by a grayscale transform and a 3×3 pixel Gaussian blur. Figure 4 shows the limb points found (every 30 images) and used in this image processing method during the simulated scenarios presented in the later sections of this paper. The x and y axes are pixel numbers in the focal plane, and the colors illustrate the time the image was taken: in chronological order from dark to light shades. It illustrates the changing Mars crescent throughout the orbit as the darker larger limbs are only quarter circles, whereas the brighter points in the center are full circles.

These limb points are extracted in a separate module, imageProcessing/LimbFinding. Given that clean images are provided to it during these simulations, the limbs are found to accurately fit the horizon. The simulation allows up to 2000 limb points to be extracted from a single image. This maximum is only approached for high-resolution (10^6 pixels and over) images in which the planet takes up a large portion of the field of view. As stated previously, the subpixel edge localization algorithm using Zernike moments would allow for better results with the same images. A method like Hough Circles could also be modified to use better edge detection algorithms.

This does require, notably for covariance analysis on the measurement, a large memory allocation that grows as N^2 (N limb points).

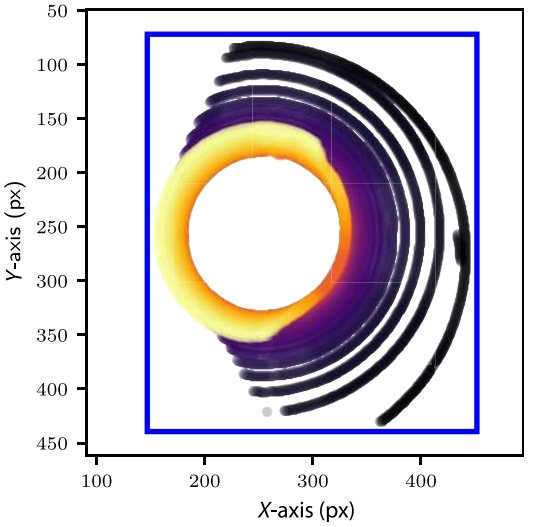


Fig. 4 Limb points found on every 30th OpNav image of Mars.

Assuming that the dense covariance is not stored and computations are done in block form, the size of the operations will depend on N. This requires a dynamic allocation, which in C requires the use of *malloc* and accesses heap memory instead of stack memory. Although not an issue in most applications, some FSW developments are reticent to use these function calls in flight.

B. Hough Circle Method

The novel method for OpNav discussed in this section uses the Hough Circle transform. This method is routinely used in robotic applications [39], where measurements are plentiful. This development attempts to apply some of these paradigms to spacecraft navigation.

The Hough transform technique follows the principle of maximum likelihood estimation, and it can thus be considered as a discretized version of a maximum likelihood estimation process [40]. Both ellipsoids and spheres project to ellipsoids on a camera plane. This has made ellipse fitting the default way to fit limbs for navigation. For ellipses, clustering methods such as Hough transforms are too numerically slow and have been discarded for navigation though discussed [41,42]. The core of the algorithm, which consists of voting for circles that the edge points detected lie upon, is summarized in Algorithm 1. Many variations exist in order to maximize speed or precision. The implementation used (named "Hough-Gradient") is an implementation of the 21HT [43,44], which provides a computationally lightweight version of the algorithm.

Algorithm 1: Baseline Hough Circle algorithm

```
image ← read(newImage)
edgePoints ← canny(image)
VoteAccumulation ← zeros
for (x,y) in edgePoints do
    for radius in [min, max] do
        for theta in [0, 2π] do
            u←x–radius cos(theta)
            v←y–radius sin(theta)
            VoteAccumulation[u,v,radius] +=1
circle ← index(max(VoteAccumulation))
return circle
```

In Basilisk, a geometrical method is used to extract pose information from center and apparent angular diameter information (pixelLineConverter [https://tinyurl.com/y2e9wcet]). The norm of the position vector is given by the apparent size, and its direction is given by the pixel and line data. Using ${}^{\mathcal{C}}\boldsymbol{r}_{\mathcal{BN}}$ as the relative vector of the camera with respect to the celestial center defined in the image plane, A as the apparent diameter of the celestial body, and D as the actual diameter,

$$|\boldsymbol{r}_{\mathcal{BN}}| = \frac{1}{2}\frac{D}{\sin[(1/2)A]} \tag{7}$$

$$^{\mathcal{C}}\tilde{\boldsymbol{r}}_{\mathcal{BN}} = [x \;\; y \;\; 1]^T \tag{8}$$

The vector $\tilde{\boldsymbol{r}}_{\mathcal{BN}}$ represents $\boldsymbol{r}_{\mathcal{BN}}$ in the image plane (it is not a unit vector). These equations have been used in multiple instances [1,45]. The vector components of $\boldsymbol{r}_{\mathcal{BN}}$ are expressed relative to the inertial frame assuming inertial attitude knowledge from the star-trackers. Using the position of the camera on the spacecraft, this provides the measurement value for an orbit determination filter using a circle-finding algorithm, though the camera position in the spacecraft body frame is negligible.

In the case of the geometric formula, the partials allow to quantify error due to the camera specifications. The following derivation uses notation common to the literature [36], and it is performed in the image plane, hence normalizing by the focal length. In this section, hats over vectors designate unit vectors, whereas the tilde signifies that the vector is in the image plane. The terms d_x, d_y are the ratios of focal length to pixels pitches; x, y are the position on the image plane in $\mathcal{C}$; u, v, ρ are the pixel values for the CAD measurements (center coordinates and radius); and u_p, v_p designates the principal point (where the camera boresight intersects the image plane). Pixels are counted starting from the upper left corner of the image, with positive u to the right in the image (pixel columns) and positive v down in the image (pixel rows). Assuming a skewness (α) in the detector,

$$^{\mathcal{C}}\tilde{\boldsymbol{r}}_{\mathcal{BN}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{d_x} & \frac{-\alpha}{d_x d_y} & \frac{\alpha v_p - d_y u_p}{d_x d_y} \\ 0 & \frac{1}{d_y} & \frac{-v_p}{d_y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{9}$$

$$\begin{aligned} |\boldsymbol{r}_{\mathcal{BN}}| &= \frac{1}{2}\frac{D}{\sin[(1/2)A]} \\ &= \frac{1}{2}\frac{D}{\sin[\arctan(\rho/d_x)]} \end{aligned} \tag{10}$$

Equations (9) and (10) are combined in order to provide the position vector estimate from the measurement in pixels:

$$\boldsymbol{r}_{\mathcal{BN}} = |\boldsymbol{r}_{\mathcal{BN}}|\frac{\tilde{\boldsymbol{r}}_{\mathcal{BN}}}{|\tilde{\boldsymbol{r}}_{\mathcal{BN}}|} = |\boldsymbol{r}_{\mathcal{BN}}|\hat{\tilde{\boldsymbol{r}}}_{\mathcal{BN}} \tag{11}$$

To map the uncertainty on the pixel measurements to the position, the total derivative of the $\boldsymbol{r}_{\mathcal{BN}}$ vector is written. This indentifies the partials that need to be computed.

$$\mathrm{d}\boldsymbol{r}_{\mathcal{BN}} = \frac{\partial \boldsymbol{r}_{\mathcal{BN}}}{\partial u}\delta u + \frac{\partial \boldsymbol{r}_{\mathcal{BN}}}{\partial v}\delta v + \frac{\partial \boldsymbol{r}_{\mathcal{BN}}}{\partial \rho}\delta\rho \tag{12}$$

Equation (9) provides an immediate partials with respect to the center measurement $^{\mathcal{C}}\boldsymbol{c}_{\mathrm{pix}} = [u \;\; v \;\; 1]^T$. In the case of no sensor skewness, this becomes

$$\frac{\partial \tilde{\boldsymbol{r}}_{\mathcal{BN}}}{\partial \boldsymbol{c}_{\mathrm{pix}}} = \begin{bmatrix} \frac{1}{d_x} & 0 & \frac{-u_p}{d_x} \\ 0 & \frac{1}{d_y} & \frac{-v_p}{d_y} \\ 0 & 0 & 1 \end{bmatrix} \quad \frac{\partial \tilde{\boldsymbol{r}}_{\mathcal{BN}}}{\partial u} = \tilde{\boldsymbol{r}}_u = \begin{bmatrix} \frac{1}{d_x} \\ 0 \\ 0 \end{bmatrix}$$

$$\frac{\partial \tilde{\boldsymbol{r}}_{\mathcal{BN}}}{\partial v} = \tilde{\boldsymbol{r}}_v = \begin{bmatrix} 0 \\ \frac{1}{d_y} \\ 0 \end{bmatrix} \tag{13}$$

where $\tilde{\boldsymbol{r}}_u$ and $\tilde{\boldsymbol{r}}_v$ are introduced to simplify notation. The partial for Eq. (10) is found by applying the chain rule, and using the definition of relevant trigonometric functions,

$$\begin{aligned} \frac{\partial |\boldsymbol{r}_{\mathcal{BN}}|}{\partial \rho} &= -\frac{1}{2}\frac{(D/d_x)}{1+(\rho/d_x)^2}\frac{\sqrt{1+(\rho/d_x)^2}}{(\rho/d_x)^2} \\ &= -\frac{D}{2}\left(\frac{d_x}{\rho}\right)^2\frac{\sqrt{d_x^2+\rho^2}}{d_x^2+\rho^2} \end{aligned} \tag{14}$$

and the partial of $\tilde{\boldsymbol{r}}_{\mathcal{BN}}/|\tilde{\boldsymbol{r}}_{\mathcal{BN}}|$ with respect to an arbitrary variable w (either u or v in our scenario) is

$$\begin{aligned} \frac{\partial \hat{\tilde{\boldsymbol{r}}}_{\mathcal{BN}}}{\partial w} &= \frac{1}{|\tilde{\boldsymbol{r}}_{\mathcal{BN}}|^2}\left(\frac{\partial \tilde{\boldsymbol{r}}_{\mathcal{BN}}}{\partial w}|\tilde{\boldsymbol{r}}_{\mathcal{BN}}| - \frac{\partial |\tilde{\boldsymbol{r}}_{\mathcal{BN}}|}{\partial w}\tilde{\boldsymbol{r}}_{\mathcal{BN}}\right) \\ &= \frac{1}{|\tilde{\boldsymbol{r}}_{\mathcal{BN}}|}\left(\frac{\partial \tilde{\boldsymbol{r}}_{\mathcal{BN}}}{\partial w} - \left(\frac{\partial \tilde{\boldsymbol{r}}_{\mathcal{BN}}}{\partial w}\cdot\hat{\tilde{\boldsymbol{r}}}_{\mathcal{BN}}\right)\hat{\tilde{\boldsymbol{r}}}_{\mathcal{BN}}\right) \end{aligned} \tag{15}$$

The normalization factors are carried in order to keep $\tilde{\boldsymbol{r}}_{\mathcal{BN}}$ in the focal plane. With these intermediate partials, the three partials required by Eq. (12) can be written out:

$$\frac{\partial \boldsymbol{r}_{\mathcal{BN}}}{\partial u} = \frac{1}{2|\tilde{\boldsymbol{r}}_{\mathcal{BN}}|}\frac{D}{\sin[\arctan(\rho/d_x)]}(\tilde{\boldsymbol{r}}_u - (\tilde{\boldsymbol{r}}_u\cdot\hat{\tilde{\boldsymbol{r}}}_{\mathcal{BN}})\hat{\tilde{\boldsymbol{r}}}_{\mathcal{BN}}) \tag{16}$$

$$\frac{\partial \boldsymbol{r}_{\mathcal{BN}}}{\partial v} = \frac{1}{2|\tilde{\boldsymbol{r}}_{\mathcal{BN}}|}\frac{D}{\sin[\arctan(\rho/d_x)]}(\tilde{\boldsymbol{r}}_v - (\tilde{\boldsymbol{r}}_v\cdot\hat{\tilde{\boldsymbol{r}}}_{\mathcal{BN}})\hat{\tilde{\boldsymbol{r}}}_{\mathcal{BN}}) \tag{17}$$

$$\frac{\partial \boldsymbol{r}_{\mathcal{BN}}}{\partial \rho} = -\frac{D}{2}\left(\frac{d_x}{\rho}\right)^2\frac{\sqrt{d_x^2+\rho^2}}{d_x^2+\rho^2}\hat{\tilde{\boldsymbol{r}}}_{\mathcal{BN}} \tag{18}$$

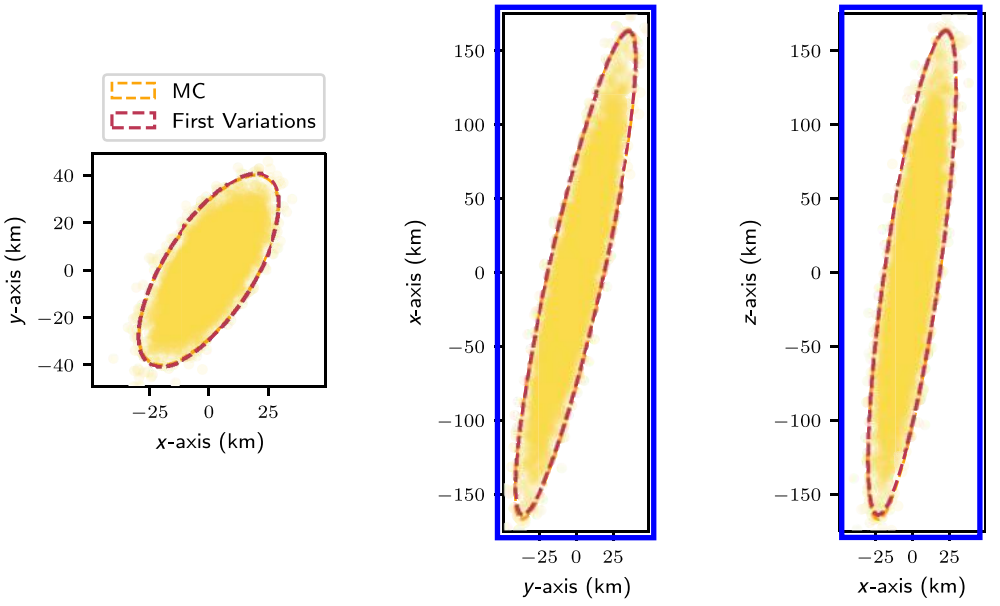
The partials in Eqs. (16–18) can be found in many forms in the literature [46] and are made explicit here in order to outline their implementation. Given that all of the key partials have been identified, the final covariance (assuming that the variables u, v, and ρ are independent) can be computed as follows:

$$\begin{aligned} \mathbb{E}\left[\delta\boldsymbol{r}_{\mathcal{BN}}\delta\boldsymbol{r}_{\mathcal{BN}}^T\right] &= \frac{\partial \boldsymbol{r}_{\mathcal{BN}}}{\partial u}\mathbb{E}\left[\delta u\delta u^T\right]\frac{\partial \boldsymbol{r}_{\mathcal{BN}^T}}{\partial u} + \frac{\partial \boldsymbol{r}_{\mathcal{BN}}}{\partial v}\mathbb{E}\left[\delta v\delta v^T\right]\frac{\partial \boldsymbol{r}_{\mathcal{BN}^T}}{\partial v} \\ &\quad + \frac{\partial \boldsymbol{r}_{\mathcal{BN}}}{\partial \rho}\mathbb{E}\left[\delta\rho\delta\rho^T\right]\frac{\partial \boldsymbol{r}_{\mathcal{BN}^T}}{\partial \rho} \\ &= \sigma_u^2\frac{\partial \boldsymbol{r}_{\mathcal{BN}}}{\partial u}\cdot\frac{\partial \boldsymbol{r}_{\mathcal{BN}^T}}{\partial u} + \sigma_v^2\frac{\partial \boldsymbol{r}_{\mathcal{BN}}}{\partial v}\cdot\frac{\partial \boldsymbol{r}_{\mathcal{BN}^T}}{\partial v} \\ &\quad + \sigma_\rho^2\frac{\partial \boldsymbol{r}_{\mathcal{BN}}}{\partial \rho}\cdot\frac{\partial \boldsymbol{r}_{\mathcal{BN}^T}}{\partial \rho} \end{aligned} \tag{19}$$

The values for σ_u, σ_v, and σ_ρ (standard deviations of the uncertainty in pixel measurements [σ_{pix} in the simulation parameters]) are driven and set by the image processing algorithm that extract the features from the image. Equation (19) is validated by Monte Carlo analysis in Figure 5. This shows 10,000 points propagated through Eq. (7) using the camera parameters in Table 1 for a range of 20,000 km with Mars offset from the image center by (100, 156) pixels. The pixel standard deviations used are $\sigma_u = \sigma_v = 0.5$ and $\sigma_\rho = 2$, similar to those seen in the literature [11].

Figure 5 shows good accordance of the first variations to the Monte Carlo. The image processing methods used here for center and apparent diameter are Hough Circle transforms [7] instantiated with the open-source computer vision library OpenCV. Given the scenario in which it is applied (orbit around a known spherical celestial body), the Hough Circle transform provides a robust solution.

Figure 6 shows every 30th circles found in the scenario using Hough Circles to fit Mars. It illustrates the stability of the attitude controller, as well as provides some high-level insight of what features were extracted. The circle that is most off-center corresponds to the start of the attitude lock. Similar to Fig. 4, the x an y axes are



a) $Y - X$ components b) $Z - Y$ components c) $Z - X$ components

Fig. 5 Matching the analytical partials with a Monte Carlo simulation in order to validate the derivation. Dotted ellipses are 3-σ bounds.

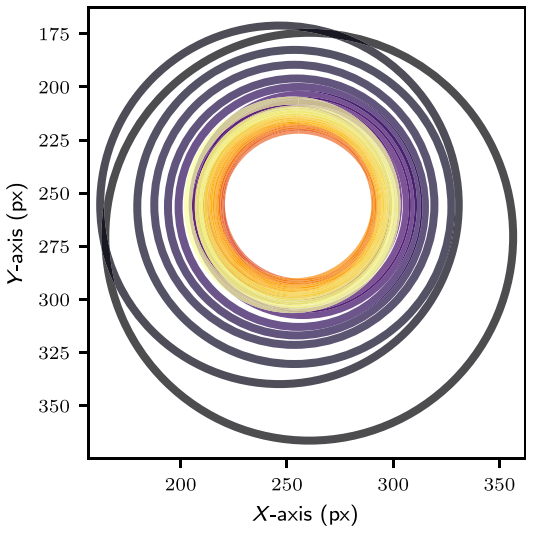


Fig. 6 Circle fits of every 30th OpNav image of Mars. Colors illustrate the order of the image capture: from dark to light shades.

pixel numbers in the focal plane, and the colors illustrate the order of the image capture: from dark to light shades.

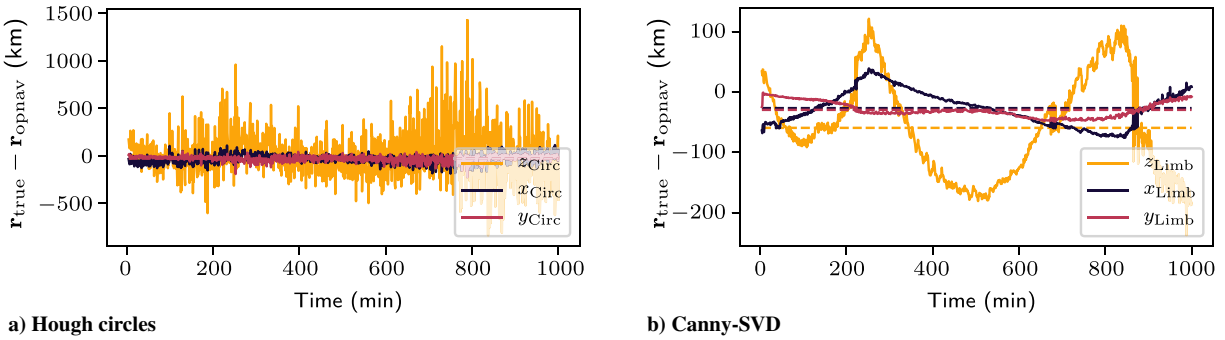
As with the previous method, FoV corrections are applied. This is done for a given point (x_d, y_d) by moving in polar coordinates —$l_d = \sqrt{x_d^2 + y_d^2}$ and $\phi = \arctan 2(y_d/x_d)$—scaling the distance from the center by $l_d/\tan(l_d)$ before moving back to Cartesian coordinates. This value is then normalized and scaled by $|\boldsymbol{r}_{\mathcal{BN}}|$ computed in Eq. (10), and rotated into the desired frames. This is done using the star-tracker estimate of $[\mathcal{BN}]$ and the known DCM from body to camera frame $[\mathcal{CB}]$.

C. Relative Orbit Determination

Now that two methods have been developed to extract measurements from images, they can be filtered in a OD estimator. The field of statistical orbit determination provides a series of potential solutions for on-board, sequential, state determination [47]. In this paper, the same filter implementation is used for both methods, with the same values for process noise:

$$[Q] = \mathrm{diag}(10^{-6}\ \mathrm{m}^2, 10^{-6}\ \mathrm{m}^2, 10^{-6}\ \mathrm{m}^2, 10^{-8}\ \mathrm{m}^2/\mathrm{s}^2, 10^{-8}\ \mathrm{m}^2/\mathrm{s}^2, 10^{-8}\ \mathrm{m}^2/\mathrm{s}^2) \quad (20)$$

The measurement noise is extracted from the images and rotated into the proper frame. Figure 7 shows the measurement quality for each of the methods by comparing the measurements directly to the truth value of the spacecraft position in the camera frame. Both of these methods show that the error in the measurements stem primarily from the ranging problem. The Z direction in the camera frame is the direction that suffers from the most errors in both methods as both algorithms present more sensitivity to a slight variation in apparent



a) Hough circles b) Canny-SVD

Fig. 7 Measurements residuals for both methods. Colors represent the vector components of the difference between the true and measured position in the camera frame.

planet size. It should be noted that the Hough Circle measurements provide more noise (seen in Fig. 7). This noise can be characterized by removing the smoothed average [48] (see Fig. 8a) and putting in histogram form in Fig. 8b.

The Shapiro–Wilk [49,50] test for normality outputs a value between 0 and 1, where 1 is a perfect Gaussian distribution. When run on this orbit data, it returns $W = 0.971$, which indicates the strongly Gaussian nature of the data. Furthermore, the mean and standard deviations experimentally found are $(0.07, 242)$ km, consistent with the plot. The noise is not highly variable over the orbit, and the test therefore suggests that it is well approximated by white noise. The same test can be done without removing the smoothed average, and it still provides a normal distribution to a high confidence with $W = 0.966$.

Figure 9 shows the errors from the limb-fitting method. A signal appears in these residuals, which can also be seen in the Hough Circles. This signal is periodic with the orbit and changes only with the lighting conditions: if the orbit is exactly the same but the light source is displaced, the signal changes periodically.

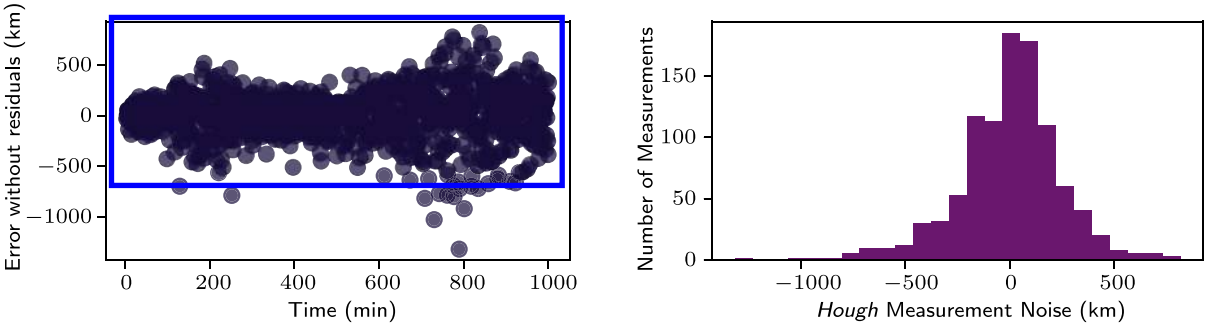
As the limbs and the circles are found using only the illuminated pixels, if these are scarce and only on a side of the planet, the measurement can suffer from a bias. The residual errors are, in both cases, not necessarily due to the OpNav transforms used, but rather to the image processing method implemented, the changing lightning, and the camera parameters. These dependencies, to be fully quantified, require in-depth sensitivity analysis, which is considered out of scope for this paper. For both methods, the pixel measurement noise value is set as a constant: $\sigma_u = \sigma_v = \sigma_\rho = \sigma_{\text{pix}} = 5.5$. This is slightly high because of the edged detection method (Canny), and the varying lighting conditions create error fluctuations that must be accounted for in order for the state errors to be within the covariance bounds. Errors stemming from fitting waxing and waning crescents have been described in Ref. [13]. A better limb-fitting method, and another algorithm to restrain the size of the limb being processed would lead to better estimates by both methods.

This filter that estimates spacecraft position and velocity in the inertial frame is implemented as a square-root unscented Kalman Filter(uKF) [51]. The filter state is

$$\boldsymbol{X} = \begin{bmatrix} {}^{\mathcal{N}}\boldsymbol{r}_{\mathcal{BN}} \\ {}^{\mathcal{N}}\dot{\boldsymbol{r}}_{\mathcal{BN}} \end{bmatrix} \tag{21}$$

where ${}^{\mathcal{N}}\boldsymbol{r}_{\mathcal{BN}}$ is the spacecraft position relative to the celestial body (Mars). The dot represents a derivative as seen by the inertial frame [31].

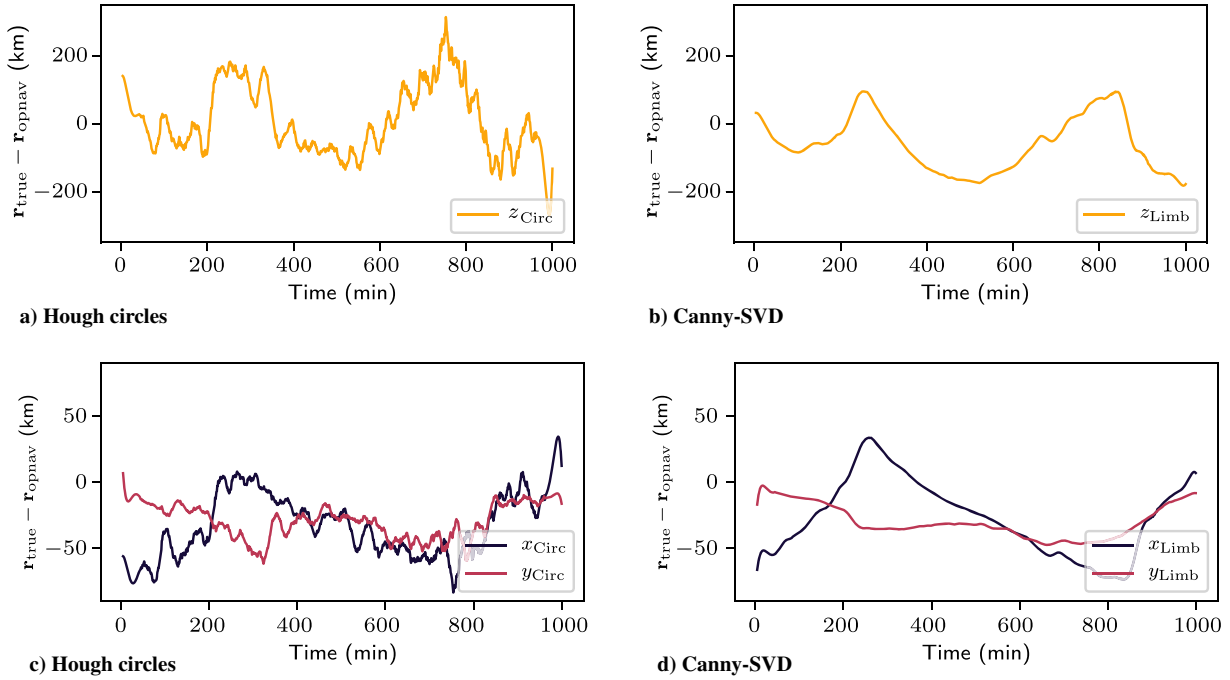
$$\dot{\boldsymbol{X}} = F(\boldsymbol{X}) = \begin{bmatrix} \dot{\boldsymbol{r}}_{\mathcal{BN}} \\ \ddot{\boldsymbol{r}}_{\mathcal{BN}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{r}}_{\mathcal{BN}} \\ -\dfrac{\mu}{|\boldsymbol{r}_{\mathcal{BN}}|^3}\boldsymbol{r}_{\mathcal{BN}} \end{bmatrix} + \boldsymbol{v} \tag{22}$$



a) Time history of measurements

b) Histogram of the measurements (25 bins)

Fig. 8 Noise about the mean Hough measurements after removing smoothed average from the raw measurement residuals in time history form (left) and histogram form (right).



a) Hough circles

b) Canny-SVD

c) Hough circles

d) Canny-SVD

Fig. 9 Measurement residuals smoothed and split between the along-track (x-y components in the camera frame) and boresight (z component) components. All smoothing is performed with a fifth-order Savitzky–Golay filter [48].

Although pointing is done in concert with the OD, the two algorithms remain decoupled, and therefore there is no attitude component to the state. The dynamics of the filter are given in Eq. (22), where $\boldsymbol{\nu}$ is a Gaussian noise term representing process noise. The state propagation is done using an Runge–Kutta fourth-order integrator. The following square-root uKF coefficients are used: $\alpha = 0.02$ and $\beta = 2$. These allow to vary the Gaussian nature of the noise. The filter does not know about the influence of other gravitational bodies. These only represent slight perturbations [47], which are not perceived given the measurement errors as well as the measurement density.

The measurement model is simple given the preprocessing done by the two methods described. This is, in practice, equivalent to extracting the measurement model from the filter code base. Therefore the measurements model in the filter is

$$G(\boldsymbol{X}) = {}^{\mathcal{N}}\boldsymbol{r}_{\mathcal{BN}} + \boldsymbol{w} \tag{23}$$

where $\boldsymbol{w}$ is a Gaussian noise term representing measurement noise. This is done to simplify the upkeep and modularity of the filter for OpNav, but also to be able to interface different types of OpNav measurements models to one same OD filter with minimal code changes.

IV. Simultaneous Attitude Guidance and Orbit Determination

This section analyses the performance of a spacecraft on an elliptical orbit around Mars. The initial conditions and simulations parameters are described and discussed in Tables 2–5. This section shows the orbit error solution convergence, as well as nominal performance of other flight-software algorithms.

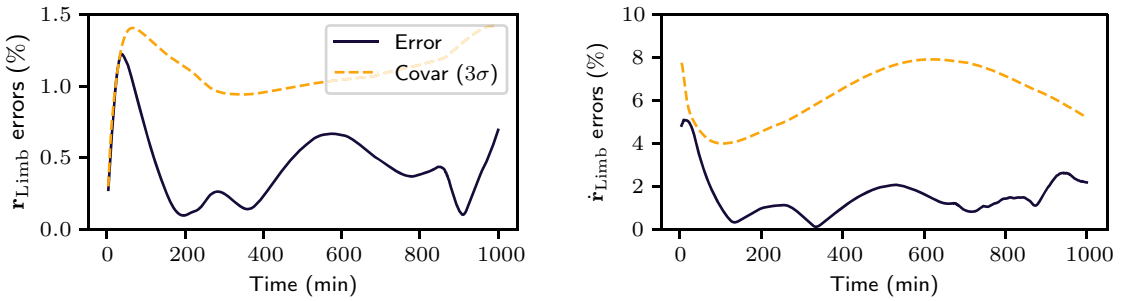
This section showcases the possible results for autonomous OpNav with a spacecraft taking numerous pictures: one image per minute. This approach also enables coupled pointing on a wide range of orbits and permits less accurate methods to perform despite noisy measurements. All of the results shown have the spacecraft perform both duties, given initial conditions that provide the planet in the field of view.

Figure 10 shows the filter results for the limb fitting method coupled with the Canny-SVD algorithm. All percentages are computed as the norm of the difference between the estimate and the truth, normalized by the truth norm, multiplied by 100. The covariance on the position components oscillates around 250 km (∼1.2%) error on position and 0.05 km/s (∼6.5%) error on velocity, with slightly better performance observed in the out-of-plane direction. The state estimates have errors in percentages below 0.5% on position and below 1.8% on velocity. These errors oscillate depending on the lighting conditions (coupled with the image-processing stack), which is seen as well in the results for the Hough Circles. In both of these results, the rising covariance on the velocity and position is driven by the elliptical nature of the orbit (true velocity decreases and relative distance increases).

The Hough Circle method performs well overall, as seen in Fig. 11. The Hough Circle covariances also range between 200 and 400 km (∼1.2%) error on position and 0.05 km/s (∼6.8%) on velocity. The state estimates errors look very similar to the limb-based method, as the measurement residuals drive the overall error trends. There is a slightly noisier signal throughout, due to the clearly noisier measurements, but the results show that this noise is closely approximated by Gaussian noise: provided enough measurements, the filter can extract a good mean state estimate. The numerical results are summarized in Table 6.

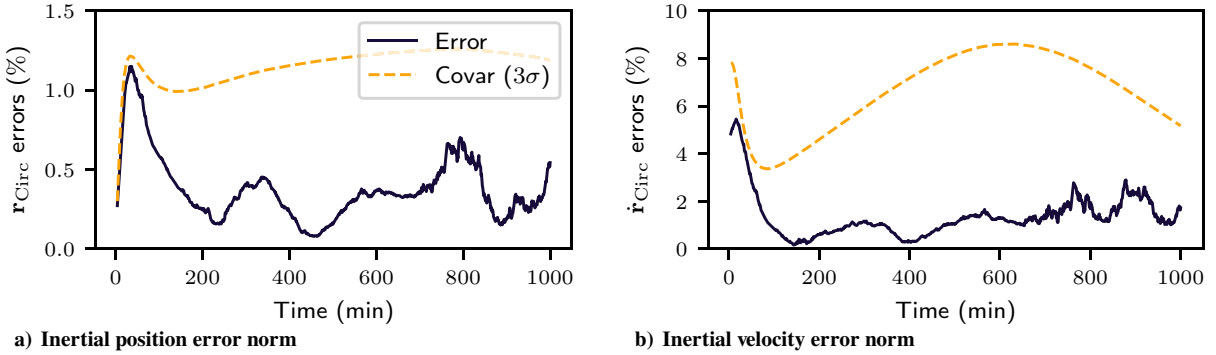
Given the measurement noise observed on the Hough Circles in Fig. 7, the fact that the new algorithm performs well could be unexpected. The analysis of this results provides an explanation: although the measurements are noisy about the residual errors shared by both methods, the noise is well characterized by Gaussian noise (see Fig. 8). Coupled with the fact that the measurements are performed frequently, this allows for the navigation filter to characterize the noise. Finally, the centering of the planet on the image plane does indeed allow for the oblateness to be negligible.

Overall, the results of both methods are consistent and promising for autonomous navigation, especially given the simplicity of the limb finding algorithm. Indeed, subpixel edge localization algorithm using Zernike moments would allow for better results with the same images for both methods.



a) Inertial position error norm b) Inertial velocity error norm

Fig. 10 Relative navigation errors: Canny–SVD.



a) Inertial position error norm b) Inertial velocity error norm

Fig. 11 Relative navigation errors: Hough Cirlces

Table 6 Summary of simulation results

Method	Position error (%)	Position covariance (%)	Velocity error (%)	Velocity covariance (%)
Canny–SVD	0.50	1.13	1.76	6.41
Hough Circles	0.43	1.16	1.63	6.81

Table 7 Monte Carlo dispersions

Variable	Distribution
Semimajor axis	$\mathcal{N}(25{,}000, 1000)$ (km)
Eccentricity	$\mathcal{U}(0.2, 0.4)$
Inclination	$\mathcal{U}(-20, 20)$ (°)
True anomaly	$\mathcal{U}(-70, 110)$ (°)
Filter position initial error	$\mathcal{N}(0, 2)$ (km)
Filter velocity initial error	$\mathcal{N}(0, 0.1)$ (km/s)
True anomaly	$\mathcal{U}(-70, 110)$ (°)

Finally, in order to ensure that the initial conditions were not outliers, 500 runs are produced with varying initial conditions according to Table 7. The results for the Hough Circle transform are shown in Fig. 12. The results, as shown previously, are consistently below 2% error on position and 6% error on velocity (consistent with the covariances seen above). The average percentage error is seen slightly increasing because of the mean orbit path as seen in the covariance for the individual run. Furthermore, some outliers are seen not finding the planet at the start (due to starting in an eclipse) and the spacecraft then tumbles in search of the planet. Some spacecraft end up finding the planet and performing OD nominally within the simulation time.

V. Outlier Mitigation

The goal of this section is to highlight the inherent robustness of the Hough Circle transform. As a clustering method, it does not require any additional algorithm (such as RANSAC) to ensure that the features extracted are as expected.

More generally, in order to prepare and validate an autonomous OpNav algorithm, analysis of outliers and their potential impact on the state estimate must be well understood. When testing and validating autonomous systems, testing for the "off-nominal" case is crucial to understanding the limits of the performances and mission feasibility. This research framework permits the addition of faulty measurements to the simulation and identification of poorly processed measurements. This is done by harnessing the flexibility and modularity of the two interfacing simulation packages. In a real mission scenario, these corrupted measurements could be weighted less favorably than trust-worthy ones when identified.

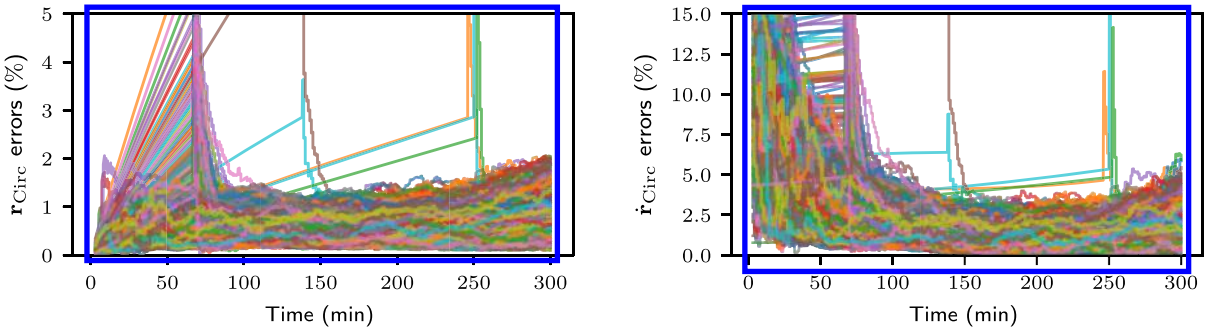
Cosmic rays are modeled in Basilisk by randomly choosing a point on the sensor as well as second point within a maximal distance of the first one. The abundance of cosmic rays on an image depends on the shutter speed among other parameters, and the module allows to toggle the frequency and quantity of such events. The outline of the model is described as follows and only depends on a single variable CR:

1) Cosmic rays are added with a probability of $p(\mathrm{CR}) = 1 - (1/\mathrm{CR}^2)$, where CR is the input parameter. The threshold is defined such that $\mathrm{CR} = 2$ provides roughly a $1/4$ chance of getting a ray, whereas $\mathrm{CR} = 10$ will generate roughly 10 rays per image. $\mathrm{CR} = 2$ provides one to two rays every other image.

2) The length of each ray is bounded by a 50×50 pixel box, and can be modified.

3) Each call to the method adds one ray, and the model attempts to add CR cosmic rays (one is only added with probability $p(\mathrm{CR})$).

In the following examples, the main setting for the image noise is $\mathrm{CR} = 3$. Such an image is illustrated in Fig. 13 (left) in which three



a) Inertial position error norm **b) Inertial velocity error norm**

Fig. 12 Monte Carlo using Hough Cirlces (500 runs).

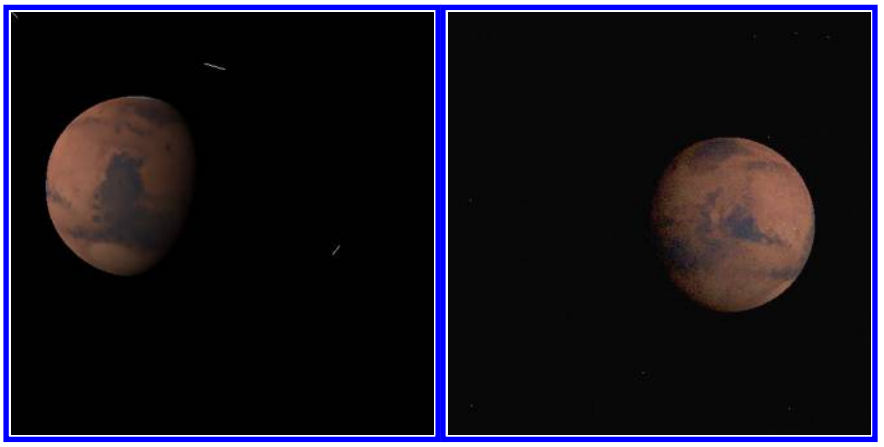


Fig. 13 Image with various corruptions. Left: cosmic rays added at random. Right: stuck and dead pixels, and blurring.

Table 8 Fault initial states

$\sigma_{\mathcal{BN}}$	$\omega_{\mathcal{BN}}$ [rad/s]	Orbital elements (a, e, i, Ω, ω, f)	Camera models
$[0\ 0\ 0]^T$	$[0\ 0\ 0]^T$	(20,000 km, 0.6, 10°, 25°, 190°, 80°)	Gaussian noise $\sigma = 5$ (pixels) CosmicRays $= 3$ Blur $\sigma = 3$ (pixels)

cosmic rays are pictured on a navigation image. At the start of the simulations, arbitrary pixels are also set on stuck on and stuck off, color and background noise are added, as well as there is a Gaussian blurring filter over the whole image. This can provide very noisy and fault-prone images. More details on the implementation of these artifacts are provided in Basilisk documentation.

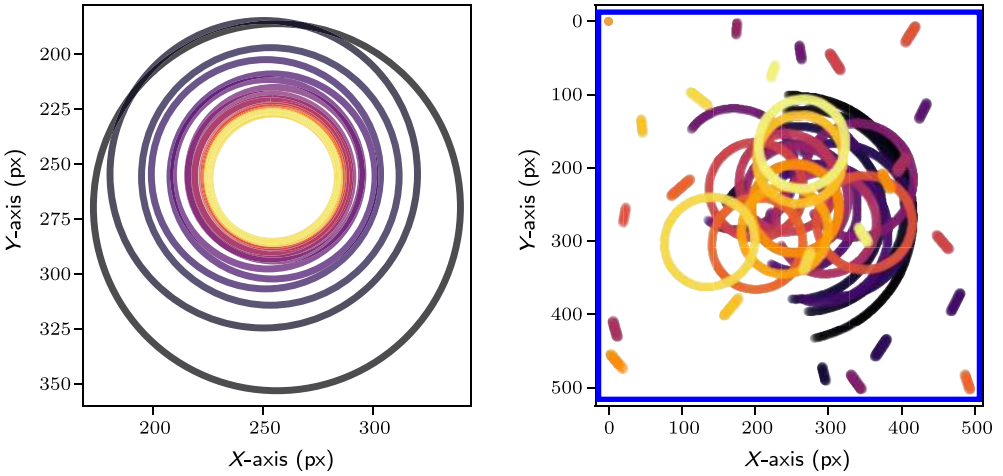
Table 8 shows the initial conditions for the fault simulation. No dynamics or previous FSW algorithms are modified. The only difference is the addition of faults.

The impact of the cosmic rays and other perturbations on the detected images can be seen in Fig. 14. To robustly track features on images, RANSAC is implemented to reject outliers. These features would not be used as is for in an SVD estimation framework as is. Nonetheless, the robustness of Hough Circles can be illustrated in this case. If the noisier performance of Hough Circles can be seen in previous results, it nevertheless provides intrinsic robustness. As a fast and lightweight algorithm, it could be used as part of another FSW stack, or on its own in order to provide a natively robust solution.

Figure 15 shows the performance of Hough Cirlces when implemented with all of the faults described. The performance of this implementation is almost indistinguishable to plots shown for either methods in ideal conditions (some extra peaks are detected). The results display the robustness of Hough Circles, for which no other parameters were modified in this section. A platform for testing arbitrary faults given a chosen mission profile is also implemented and provided to the community.

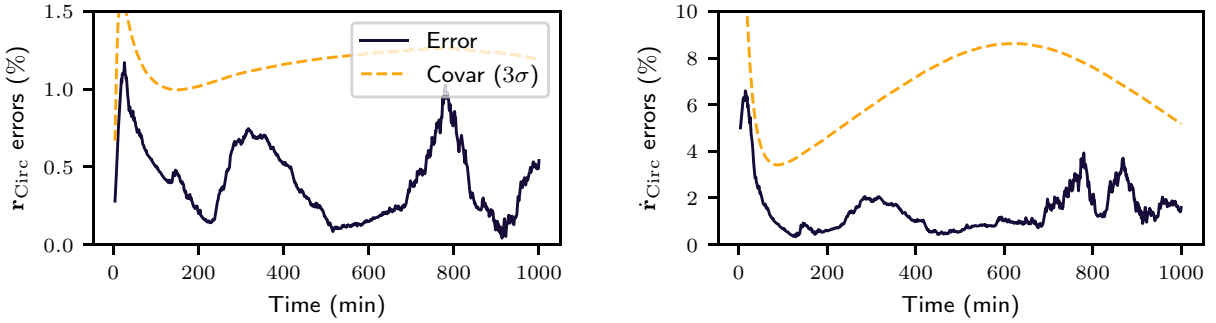
VI. Conclusions

This paper presents novel developments on several fronts. It provides estimation results using solely autonomous OpNav on orbit about Mars, in order to quantify the accuracy achieved with different methods. It also introduces a Hough Circle–based navigation method. Although this method does not perform to the same level as current state-of-the-art algorithms, it provides a robust alternative with single algorithm implementation. The difference between state-of-the-art algorithms and the new Hough Circles method is exemplified throughout the paper, acknowledging that better results are possible with improved edge detection algorithms. A core delineation lies in the choice of fitting ellipses or circles: yet a simple circle fit with a spacecraft aiming to center the planet on the camera plane has shown good results given the camera resolution. Finally the entire study is done with an underlying pointing coupling with the orbit determination. Doing simultaneous pointing and OD adds a level of fidelity to the simulation presented. Furthermore, it supports the development of a high-image count OpNav framework. By taking pictures frequently (every minute), even a less accurate method like Hough Circles shows valid results for autonomous navigation around Mars. Finally, this work includes fault mitigation examples, and displays the inherent robustness of the Hough clustering method.



a) Hough circles with perturbations b) Cannyedge detection with perturbations

Fig. 14 Image processing in case of faults: CR = 2. The limb finding does not use any robust clustering method in order to illustrate the faults.



a) Hough circles b) With mitigation in fault mode 2

Fig. 15 State estimation with fault detection.

Acknowledgment

The authors would like to acknowledge the continuous help of Jennifer Wood, Samuel Bateman, and Scott Piggott.

References

[1] Owen, W. M., "Optical Navigation Program Mathematical Models," Jet Propulsion Lab. Engineering Memorandum 314-513, Aug. 1991.

[2] Bhaskaran, S., Riedel, J., Synnott, S., and Wang, T., "The Deep Space 1 Autonomous Navigation System—A Post-Flight Analysis," AIAA Paper 2000-3935, 2000.
https://doi.org/10.2514/6.2000-3935

[3] Kubitschek, D. G., "Deep Impact Autonomous Navigation: The Trials of Targeting the Unknown," *29th Annual AAS Guidance and Control Conference, Breckenridge*, Jet Propulsion Laboratory, NASA, Pasadena, CA, 2006, pp. 1–3.

[4] Cheng, Y., Goguen, J., Johnson, A., Leger, C., Matthies, L., Martin, M. S., and Willson, R., "The Mars Exploration Rovers Descent Image Motion Estimation System," *IEEE Intelligent Systems*, Vol. 19, No. 3, 2004, pp. 13–21.
https://doi.org/10.1109/MIS.2004.18

[5] Martin, A. M. S., Bayard, D. S., Conway, D. T., Mandic, M., and Bailey, E. S., "A Minimal State Augmentation Algorithm for Vision-Based Navigation Without Using Mapped Landmarks," *GNC 2017: 10th International ESA Conference on GNC Systems*, Vol. 10, 2017.

[6] Carson, J. M., Seubert, C., Amzajerdian, F., Bergh, C., Kourchians, A., Restrepo, C., Villalpando, C. Y., O'Neal, T., Robertson, E. A., Pierrottet, D. F., Hines, G. D., and Garcia, R., "COBALT: Development of a Platform to Flight Test Lander GN&C Technologies on Suborbital Rockets," AIAA Paper 2017-1496, 2017.
https://doi.org/10.2514/6.2017-1496

[7] Petkovic, T., and Loncaric, S., "An Extension to Hough Transform Based on Grandient Orientation," *Proceedings of the Croatian Computer Vision Workshop*, Univ. of Zagreb Faculty of Electrical Engineering and Computing, Zagreb, Croatia, 2015.

[8] Weismuller, T., Caballero, D., and Leinz, M., "Technology for Autonomous Optical Planetary Navigation and Precision Landing," AIAA Paper 2007-6173, 2007.
https://doi.org/10.2514/6.2007-6173

[9] Wokes, D., and Wokes, S., "Surveying and Pose Estimation of a Lander Using Approximative Crater Modelling," AIAA Paper 2010-8342, 2010.
https://doi.org/10.2514/6.2010-8342

[10] Teil, T., Bateman, S., and Schaub, H., "Closed-Loop Software Architecture for Spacecraft Optical Navigation and Control Development," *Journal of Astronautical Sciences*, Vol. 67, 2020, pp. 1575–1599.
https://doi.org/10.1007/s40295-020-00216-1

[11] Christian, J. A., and Robinson, S. B., "Noniterative Horizon-Based Optical Navigation by Cholesky Factorization," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 12, 2016, pp. 2757–2765.
https://doi.org/10.2514/1.G000539

[12] Ying-Dong, Q., Cheng-Song, C., San-Ben, C., and Jin-Quan, L., "A Fast Subpixel Edge Detection Method Using Sobel-Zernike Moments Operator," *Image and Vision Computing*, Vol. 23, No. 1, 2005, pp. 11–17.
https://doi.org/10.1016/j.imavis.2004.07.003

[13] Hollenberg, C. L., Christian, J. A., Bhaskaran, S., and Owen, W. M., "Centroiding Performance for Horizon-Based Optical Navigation with Cassini Images of Dione and Rhea," *29th AAS/AIAA Space Flight Mechanics Meeting*, AAS Paper 19-494, San Diego, CA, 2019.

[14] Canny, J., "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 6, 1986, pp. 679–698.
https://doi.org/10.1109/TPAMI.1986.4767851

[15] Alcorn, J., and Schaub, H., "Simulating Attitude Actuation Options Using the Basilisk Astrodynamics Software Architecture," *67th International Astronautical Congress*, Paper IAC-16-C1.1.4, International Astronautical Federation (IAF), 2016.

[16] Kenneally, P. W., Piggott, S., and Schaub, H., "Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework," *7th International Conference on Astrodynamics Tools and Techniques (ICATT)*, DLR, Oberpfaffenhofen, Germany, 2018.
https://doi.org/10.2514/1.I010762

[17] Wood, J., Margenet, M. C., Kenneally, P., Schaub, H., and Piggott, S., "Flexible Basilisk Astrodynamics Visualization Software Using the Unity Rendering Engine," *AAS Guidance and Control Conference*, AAS Paper 18-093, San Diego, CA, 2018.

[18] Kenneally, P. W., and Schaub, H., "High Geometric Fidelity Modeling of Solar Radiation Pressure Using Graphics Processing Unit," *AAS/AIAA Spaceflight Mechanics Meeting*, AAS Paper 16-500, San Diego, CA, 2016, pp. 2577–2587.

[19] Kenneally, P. W., and Schaub, H., "Modeling Solar Radiation Pressure with Self-Shadowing Using Graphics Processing Unit," *AAS Guidance, Navigation and Control Conference*, AAS Paper 17-127, San Diego, CA, 2017.

[20] Kenneally, P. W., and Schaub, H., "Parallel Spacecraft Solar Radiation Pressure Modeling Using Ray-Tracing on Graphic Processing Unit," *International Astronautical Congress*, International Astronautical Federation (IAF), Paper IAC-17,C1,4,3,x40634, 2017.

[21] Alcorn, J., Allard, C., and Schaub, H., "Fully Coupled Reaction Wheel Static and Dynamic Imbalance for Spacecraft Jitter Modeling," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 6, 2018, pp. 1380–1388.
https://doi.org/10.2514/1.G003277

[22] Allard, C., Schaub, H., and Piggott, S., "General Hinged Solar Panel Dynamics Approximating First-Order Spacecraft Flexing," *Journal of Spacecraft and Rockets*, Vol. 55, No. 5, 2018, pp. 1291–1299.
https://doi.org/10.2514/1.A34125

[23] Cappuccio, P., Allard, C., and Schaub, H., "Fully-Coupled Spherical Modular Pendulum Model to Simulate Spacecraft Propellant Slosh," *AAS/AIAA Astrodynamics Specialist Conference*, AAS Paper 18-224, San Diego, CA, 2018.

[24] Allard, C., Diaz-Ramos, M., and Schaub, H., "Spacecraft Dynamics Integrating Hinged Solar Panels and Lumped-Mass Fuel Slosh Model," *AIAA/AAS Astrodynamics Specialist Conference*, AIAA Paper 2016-5684, 2016.

[25] Tegmark, M., "An Icosahedron-Based Method for Pixelizing the Celestial Sphere," *Astrophysical Journal Letters*, Vol. 470, No. 2, 1996, pp. L81–L84.
https://doi.org/10.1086/310310

[26] Psiaki, M., "Autonomous Lunar Orbit Determination Using Star Occultation Measurements," AIAA Paper 2007-6657, 2007.

[27] Park, W., and Jung, Y., "Robust Crater Triangle Matching Algorithm for Planetary Landing Navigation," *Jounral of Guidance, Control, and Dynamics*, Vol. 42, No. 2, 2019, pp. 402–410.
https://doi.org/10.2514/1.G003400

[28] Gaskell, R., "Optical Navigation Near Small Bodies," *21st AAS/AIAA Spaceflight Mechanics Meeting; 2011, Advances in Astronautical Sciences*, Vol. 140, Univelt, San Diego, CA, 2011, pp. 1705–1718.

[29] Riedel, J. E., Vaughan, A. T., and Werner, R. A., "Optical Navigation Plan and Strategy for the Lunar Lander Altair; OpNav for Lunar and Other Crewed and Robotic Exploration Applications," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2010-7719, 2010.
https://doi.org/10.2514/6.2010-7719

[30] Gaskell, O. B. R., and Kahn, E., "Assessing the Quality of Topography from Stereo-Photoclinometry," Univ. of Helsinki Helsinki, Finland, 2014.

[31] Schaub, H., and Junkins, J. L., *Analytical Mechanics of Space Systems*, 3rd ed., AIAA Education Series, AIAA, Reston, VA, 2014, Chap. 1.
https://doi.org/10.2514/4.102400

[32] Allard, C., Diaz-Ramos, M., Kenneally, P., Schaub, H., and Piggott, S., "Modular Software Architecture for Fully-Coupled Spacecraft Simulations," *Journal of Aerospace Information Systems*, Vol. 15, No. 12, 2018, pp. 670–683.
https://doi.org/10.2514/1.I010653

[33] Christian, J., and Robinson, S. B., "Observations of the Geometry of Horizon-Based Optical Navigation," AAS Paper 16-151, San Diego, CA, 2016.

[34] Mortari, D., D'Souza, C. N., and Zanetti, R., "Image Processing of Illuminated Ellipsoid," *Journal of Spacecraft and Rockets*, Vol. 53, No. 3, 2016, pp. 448–456.
https://doi.org/10.2514/1.A33342

[35] Christian, J. A., "Optical Navigation Using Iterative Horizon Reprojection," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 5, 2016, pp. 1092–1103.
https://doi.org/10.2514/1.G001569

[36] Christian, J., "Accurate Planetary Limb Localization for Image-Based Spacecraft Navigation," *Journal of Spacecraft and Rockets*, Vol. 54, No. 3, 2017, pp. 708–730.
https://doi.org/10.2514/1.A33692

[37] Hughes, C., Denny, P., Jones, E., and Glavin, M., "Accuracy of Fish-Eye Lens Models," *Applied Optics*, Vol. 49, No. 17, 2010, pp. 3338–3347.
https://doi.org/10.1364/AO.49.003338

[38] Hikes, J., Liounis, A. J., and Christian, J. A., "Parametric Covariance Model for Horizon-Based Optical Navigation," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 1, 2017, pp. 170–178.
https://doi.org/10.2514/1.G000708

[39] Suligoj, F., Sekoranja, B., Svaco, M., and Jerbic, B., "Object Tracking with a Multiagent Robot System and a Stereo Vision Camera," *Procedia*

Engineering, Vol. 69, Jan. 2014, pp. 968–973.
https://doi.org/10.1016/j.proeng.2014.03.077

[40] Zhang, Z., "Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting," *Image and Vision Computing*, Vol. 15, No. 1, 1997, pp. 59–76.
https://doi.org/10.1016/S0262-8856(96)01112-2

[41] Li, S., Lu, R., Zhang, L., and Peng, Y., "Image Processing Algorithms for Deep-Space Autonomous Optical Navigation," *Journal of Navigation*, Vol. 66, No. 4, 2013, pp. 605–623.
https://doi.org/10.1017/S0373463313000131

[42] Christian, J., "Onboard Image-Processing Algorithm for a Spacecraft Optical Navigation Sensor System," *Journal of Spacecraft and Rockets*, Vol. 49, No. 2, 2012, pp. 337–352.
https://doi.org/10.2514/1.A32065.

[43] Yuen, H., Princen, J., Illingworth, J., and Kittler, J., "Comparative Study of Hough Transform Methods for Circle Finding," *Image and Vision Computing*, Vol. 8, No. 1, 1990, pp. 71–77.
https://doi.org/10.1016/0262-8856(90)90059-E

[44] Davies, E., "A Modified Hough Scheme for General Circle Location," *Pattern Recognition Letters*, Vol. 7, No. 1, 1988, pp. 37–43.
https://doi.org/10.1016/0167-8655(88)90042-6.

[45] Battin, R. H., *An Introduction to the Mathematics and Methods of Astrodynamics*, revised ed., AIAA, Reston, VA, 1999, Chap. 14.
https://doi.org/10.2514/4.861543

[46] Owen, W., "Methods of Optical Navigation," *Proceedings of the AAS/AIAA Space Flight Mechanics Meeting*, AAS Paper 11-215, San Diego, CA, Feb. 2011.

[47] Schutz, B., Tapley, B. S., and Born, G. H., *Statistical Orbit Determination*, Elsevier Academic Press, Burlington, MA, 2004, Chap. 4.

[48] Savitzky, A., and Golay, M. J. E., "Smoothing and Differentiation of Data by Simplified Least Squares Procedures," *Analytical Chemistry*, Vol. 36, No. 8, 1964, pp. 1627–1639.
https://doi.org/10.1021/ac60214a047

[49] Shapiro, S. S., and Wilk, M. B., "An Analysis of Variance Test for Normality (Complete Samples)," *Biometrika*, Vol. 52, Nos. 3–4, 1965, pp. 591–611.
https://doi.org/10.1093/biomet/52.3-4.591

[50] Mohd Razali, N., and Yap, B., "Power Comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling Tests," *Journal of Statistical Modeling and Analytics*, Vol. 2, No. 1, 2011, pp. 21–33.

[51] Van der Merwe, R., and Wan, E. A., "The Square-Root Unscented Kalman Filter for State and Parameter-Estimation," *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, Vol. 6, IEEE, New York, 2001, pp. 3461–3464.
https://doi.org/10.1109/ICASSP.2001.940586

O. Abdelkhalik
Associate Editor