CENTROID AND APPARENT DIAMETER OPTICAL NAVIGATION ON MARS ORBIT
Thibaud Teil1, Hanspeter Schaub2 and Daniel Kubitschek3; 1 Graduate Student, Aerospace Engineering Sciences, University of Colorado, 2 Glenn L. Murphy Chair of Engineering, Department of Aerospace Engineering Sciences, University of Colorado, 3 Laboratory for Atmospheric and Space Physics, University of Colorado.

Abstract. Extensive testing of autonomous systems — on the ground and in flight — builds trust in the desired performance of the algorithms. Flight tests, or technology demonstrators, are rare and also require rigorous testing in order to not endanger the main mission. Additional tests can be achieved primarily through simulated environments. This paper describes novel results achieved for simulated autonomous optical navigation on orbit about Mars. The coupled nature of the simulation enables simultaneous pointing and orbit determination with dynamic image generation. Navigation is done solely using optical images, and by means of limb or centroid/diameter extraction. This is applicable on a wide range of orbits depending on the camera parameters. Through the implementation of pre-existing algorithms and the development of novel optical navigation methods, a fault detection capability is also introduced in order to test methods in off-nominal cases. This research provides insight into achievable navigation accuracy and image processing methods, as well as outlier detection and mitigation for mission readiness.

Introduction. Optical navigation (OpNav) in astrodynamics refers to the use of images taken by an onboard 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.

Aside from optical measurements, other autonomous navigation methods are in development. X-ray navigation using pulsars to estimate the spacecraft’s position2 was recently flown on the International Space Station (ISS).3 The Deep Space Atomic Clock (DSAC),4 is also being tested to permit one-way radio-frequency measurements sent from the Deep Space Network (DSN). Finally, StarNAV5 aims to develop a new sensor system to detect relativistic perturbations in the wavelength and direction of observed stars to measure spacecraft velocity. Although just one in many methods, this research focuses on enhancements to autonomous OpNav for spacecraft state estimation. This has also seen developments and is notably planned for use on Orion’s Exploration Mission 1,6 as well as for Jupiter and Saturn exploration using the planet satellites.7 Autonomous OpNav remains a sought-after navigation method as it requires only cameras (which can be both light-weight and used for other purposes) and fundamentally relies on imaging the object that is being studied and orbited as opposed to Earth-based data, or distant pulsars.

Past missions such as Deep Space 1,8 Stardust, and Deep Impact9 relied heavily on OpNav. Nevertheless, only Deep Space 1 and Deep Impact used autonomous OpNav (AutoNav8): as a technology experiment and mission enabler respectively. Deep Space 1 used a first implementation of AutoNav in order to determine it’s orbit within the solar system, while Deep Impact used another implementation of AutoNav in order to ensure contact with the asteroid Temple 1. The first implementation uses beacons (planets and certain stars) in order to triangulate its location. The second version used a center of brightness algorithm in order to instruct the guidance algorithms on the proper impact trajectory. 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 DIMES10 system used to land Mars rovers. These use cross-correlation methods11,12 developed specifically for EDL. These missions are examples of using autonomy as a mission enabler: the goals would otherwise not be attainable due to round-trip light time.
Routine autonomy 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. The use of such algorithms requires commensurately thorough analysis in order to develop confidence in the long-term performance. One reason that makes this difficult to achieve is 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. In order for such a simulation to provide the analysis tools required, it must enable Monte-Carlo (MC) capabilities as well as easy scriptability to modify scenarios with ease. Furthermore developing open-source code bases enables continuous validation, testing, and community support.

The example scenario referenced in this research is a spacecraft on Mars orbit using only Center and Apparent Diameter (CAD) for OpNav. This scenario — pictured in Fig. 1 — enables testing of a full OpNav sequence, from taking images to control. Building on previous work in Reference, exploring closed-loop optical navigation simulations, this paper develops a novel autonomous on-orbit OpNav architecture using Centroid and Apparent Diameter (CAD) measurements about Mars for simultaneous pointing and Orbit Determination (OD).

This work presents an implementation of a state-of-the-art limb-based OpNav method, while simultaneously using a more basic yet robust Hough Circle finding algorithm. This research presents fully coupled Attitude-Orbit OpNav on orbit, as well as comparative capabilities for different methods in a flight-like simulated environment. This is key, not only for determining what level of complexity needs to be implemented on board for a specific requirement to be satisfied, but also in order to run several methods in parallel. This is useful to develop technology tests in flight where two methods can run side by side with one more trusted method checking for faults. It also provides a framework for fault detection analysis in which several methods run simultaneously in order to harness the strengths of each implementation. This is illustrated in the last section of this paper where a fault is introduced to the images while both methods are attempting to output measurements.

Simulating a Mars OpNav Orbiter. 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; and a sister software package — Vizard — to dynamically visualize the simulation environment.

Basilisk* is a highly modular astrodynamics simulation framework that allows for the rapid simulation of complex spacecraft dynamics. Key features include solar radiation pressure, imbalanced reaction wheels, imbalanced control moment gyrosopes, flexible solar panels, fuel slosh, depletable mass, as well as multiple body gravity and gravitational spherical harmonics. The sensor simulation and actuator components couple with the spacecraft dynamics through a publish-subscribe (pub-sub) messaging system. A state engine generates complex spacecraft dynamics without having to develop and code any dynamics differential equations. 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.

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. Point distribution methods are some of many feature tracking methods which provide promising results. They can be extracted. Point distribution methods are some of many feature tracking methods which provide promising results. They come at a computation cost. Instead, the current state of the art for OpNav is Stereo-Photoclinometry (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. With the goal of autonomy, this research will focus on on-board methods for image processing.

Although there is potential for using more computationally intense methods on-board, this research focuses on implementing CAD 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. This is seen in Figure 2 where the image processing extracts center and apparent diameter (or limb points in the horizon-based navigation method). These points are then transformed into a spacecraft position

*https://hanspeterschaub.info/bskMain.html
measurement through the measurement model of the filter. The noise is also transformed to the proper frame to provide measurement noise to the filter. The estimator can then reconstruct the spacecraft states.

Figure 2 also pictures the coupled nature of the simulation: the images processed are used for attitude guidance and control, which then generate new images.

**Synthetic Images.** The images are rendered in a Unity camera simulator in realistic lighting conditions using its integrated GPU ray-tracing capability. Camera specifications, such as resolution and focal length, are generated in the camera module as well. The simulation also provides corruption capabilities through a Basilisk camera module in order to render faulty images.

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 way. The difficulty comes in providing speeds for several-orbit simulations that tightly couple in attitude variations for image generation. The simulation with a 0.5 second time-step taking 512 × 512 every minute for attitude control, alongside dynamics models, flight software algorithms, and OpNav OD algorithms during a 10 hour long orbit approximately one minute (∼ 600× real-time).

### Table 1. Camera Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma_{BS} )</td>
<td>([0, 0, 0]^T )</td>
</tr>
<tr>
<td>( B_{CBS} ) [m]</td>
<td>([0.2, 0.2]^T )</td>
</tr>
<tr>
<td>Resolution (pixels)</td>
<td>([512, 512]^T )</td>
</tr>
<tr>
<td>Sensor Size (mm)</td>
<td>([10, 10]^T )</td>
</tr>
<tr>
<td>Field of View (°)</td>
<td>([40, 40]^T )</td>
</tr>
</tbody>
</table>

Throughout this paper, the camera frame is noted \( C \), the spacecraft body frame is \( B \), while the inertial frame is \( N \). Direction cosine matrices are noted \( [BAN] \) to represent the rotation from the inertial frame to the body frame which can also be represented a Modified Rodrigues Parameters (MRPs) \( \sigma_{BN} \). The rotation rate of the body frame relative to the inertial frame is noted \( \omega_{BN} \) and the left superscript notation indicates the frame a vector is expressed in.\(^{36}\) Positions are noted \( r \), subscript \( CBS \) represents the camera position relative to the spacecraft center of the frame \( B \), similarly, \( r_{BN} \) is the vector from the center of \( N \) to the spacecraft body frame.

The camera parameters are given in Table 1 and use a square image with a wide field of view. The sensor size of 1cm is, equivalent to choosing a focal length of 1.373cm. The position and orientation of the camera are arbitrary in this scenario, as long as they don’t create any self-shadowing. This is avoided easily with the parameters implemented.

**Simulated Astrodynamics.** The scenario simulates a spacecraft on an elliptical orbit around Mars. This is chosen in order to test an imaging requirement of the planet, and to allow for variation in the apparent size of the planet. This section specifies the simulation parameters as well as the flight-software algorithms used.

The simulation uses \emph{Spice} data, where the simulation begins on December 12th at 22:00 (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. This allows the FSW algorithms to be tested along a wide variety of lighting conditions and planet sizes. The simulation modules assigned to modeling the spacecraft dynamics and environment are described in Table 3. These modules simulate spacecraft attitude gyroscopics and gravity,\(^{29}\) eclipse, reaction wheels,\(^{23}\) and star trackers. Eclipseing is also modeled: this usually creates a halt in the spacecraft measurements. In a fully coupled Attitude-OD simulation this means 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 goes through a 3 minute guided pointing mode before starting to navigate and point using OpNav.

Beyond the pure simulation 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 (Table 5, 4). The imaging modules encompass the OpNav raw measurements (limbs and circles) as well as the measurement models to provide spacecraft position.

Table 5 shows the modules implemented for centroid-based pointing guidance,\(^{15}\) OD, and control using MRP-Feedback.\(^{36}\)

**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.

**Limb Detection.** In recent years, with interest in applying autonomous navigation around the Moon,\(^{37}\) high-fidelity image processing algorithms have been developed. With the primary application to spherical or ellipsoidal

---

\(^{1}\)naif.jpl.nasa.gov/naif/
bodies, these revolve mostly around finding an ellipse centroid,38 or fitting a limb.39 These methods provide accurate measurements of a spacecraft’s relative position to the body provided an unambiguous limb.

The baseline method chosen to measure spacecraft position is the Non-Iterative Horizon-Based Optical Navigation by Singular Value Decomposition14,40 (NIH-SVD). This method was chosen for its high performance not only analytically 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 a set of \( N \) limb point vectors \( \mathbf{s}_k = (x_k, y_k, 1) \) (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 \( k^{th} \) limb point in the camera frame.

\[
\begin{align*}
[B] &= [Q_m][CP] \quad (1) \\
\mathbf{s}_k &= [B]\mathbf{s}_k \quad (2) \\
\mathbf{s}_k' &= \frac{\mathbf{s}_k}{||\mathbf{s}_k||} \quad (3)
\end{align*}
\]

Where \( [Q_m] = \text{diag}(\frac{1}{r_a}, \frac{1}{r_b}, \frac{1}{r_c}) \) in the planet-fixed frame, with \( r_a, r_b, r_c \) are the radii of the potentially ellipsoidal body along it’s principal axes. \([B] = [Q_m][CP]\), where \( P \) is the planet frame. In this paper the planet frame is taken as the inertial frame \( N \), and \([Q_m]\) represents a circular mars with \( r_a = r_b = r_c = 3396.19 \text{km} \).

After rotation and normalization according to Equations , these are concatenated in order to solve for

\[
[H] = \begin{bmatrix}
[s_0'] \\
\vdots \\
[s_N']
\end{bmatrix}
\]

\([H] \mathbf{n} = 1_{N\times1} \quad (5)\]

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 \([R_H] \mathbf{n} = [Q_H]'1_{N\times1} \) which is solved by back-substitution. With \( \mathbf{n} \) now given, the spacecraft position is given in the camera frame by:

\[
C_{\hat{r}_{BN}} = -(\mathbf{n}^T \mathbf{n} - 1)^{-\frac{1}{2}}[B]^{-1} \mathbf{n} \quad (6)
\]

The only computation left is to rotate into the desired frames. This is done using the star-tracker estimate of

<table>
<thead>
<tr>
<th><strong>Table 3. Simulation Parameters</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simulation Modules Instantiated</strong></td>
</tr>
<tr>
<td>Spacecraft Hub</td>
</tr>
<tr>
<td>Gravity Effector/Eclipse</td>
</tr>
<tr>
<td>Simple Navigation Star Tracker</td>
</tr>
<tr>
<td>Reaction Wheel Effector</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Table 4. Flight Software for Imaging</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flight Software Modules Instantiated</strong></td>
</tr>
<tr>
<td>Image Processing (arguments for Hough Circle method)</td>
</tr>
<tr>
<td>Limb Finding (arguments for Canny transform)</td>
</tr>
<tr>
<td>Pixel Line Transform</td>
</tr>
<tr>
<td>Horizon Nav</td>
</tr>
</tbody>
</table>

**Figure 3. Every 30\(^{th}\) Mars limb fit**
Necessary parameters at initialization

\[
\begin{align*}
\alpha &= 0.02, \beta = 2, \kappa = 0, \text{noiseSF} = 10 \\
r_{\text{error}} &= [10, 10, -10]\text{km}, r_{\text{error}} = [0, 0, 0.01]\text{km/s} \\
K &= 3.5, P = 30 \text{(no integral feedback)} \\
\minAngle &= 0.001^\circ, \text{timeOut} = 100\text{s} \\
\omega_{\text{search}} &= [0.06, 0.0, -0.06]^\circ/\text{s}, \tilde{\sigma}_c = [0.0, 0.01]\text{m} \\
r_{\text{error}} &= [10, 10, -10]\text{km} \\
\end{align*}
\]

\section*{Necessary parameters at initialization}

\begin{itemize}
\item \text{minAngle} = 0.001^\circ, \text{timeOut} = 100\text{s}
\item \text{\omega}_{\text{search}} = [0.06, 0.0, -0.06]^\circ/\text{s}, \tilde{\sigma}_c = [0.0, 0.01]\text{m}
\item \text{Control axes are} \{b_1, b_2, b_3\}
\end{itemize}

\begin{table}[h]
\centering
\caption{Flight Software Pointing and Orbit Determination}
\begin{tabular}{|l|l|}
\hline
Flight Software Modules Instantiated & Necessary parameters at initialization \\
\hline
OpNav Point & \minAngle = 0.001^\circ, \text{timeOut} = 100\text{s} \\
relativeOD & \omega_{\text{search}} = [0.06, 0.0, -0.06]^\circ/\text{s}, \tilde{\sigma}_c = [0.0, 0.01]\text{m} \\
MRP Feedback RW & \text{Control axes are} \{b_1, b_2, b_3\} \\
RW motor Torque & \text{K} = 3.5, P = 30 \text{(no integral feedback)} \\
\hline
\end{tabular}
\end{table}

[BN] and the known camera frame [CB], and the module outputs the covariance and estimates in the body, inertial, and camera frames for downstream use. This method is implemented in C (under the Basilisk module name of \texttt{imageProcessing/HorizonNav}). The last thing to do in order to implement this method fully is to create a limb finding method. This is done by using \texttt{Canny} transform\textsuperscript{14} implemented in \texttt{OpenCV}\textsuperscript{3}\textsuperscript{11}, preceded by a greyscale transform and a 3 \times 3 pixel Gaussian blur. Figure 3 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, while the brighter points in the center are full circles.

These limb points are extracted in a separate module \texttt{imageProcessing/LimbFinding}. Given the clean images provided to it during these simulations, the limbs found accurately hug the planet. The simulation allows up to 2000 limb points to be extracted from a single image. This maximum is only approached for high resolution (10\textsuperscript{6} pixels and over) images in which the planet takes up a large portion of the field of view.

This does require, notably for covariance analysis on the measurement, a large memory allocation which grows as \(N^2\) (N limb points). This is dynamically allocated, which in C requires the use of \texttt{malloc} and therefore 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. This is because of the potential memory leaks that can occur if heap memory is not freed, and from the potential hazard of having erroneous inputs lead to requests for more memory than can be provided by the hardware.

\textit{Hough Circle Detection.} The novel method for OpNav discussed in this section uses the \textit{Hough Circle} transform. This method is routinely used in robotic applications\textsuperscript{12} where measurements are plentiful. This development attempts to apply some of these paradigms to spacecraft navigation.

In the \textit{Hough Circle} implementation, a geometrical method is used to extract pose information from center and apparent diameter information. The norm of the position vector is given by the apparent size, its direction is given by the pixel and line data. Using \(\mathbf{r}_{\text{BN}} = [r_1, r_2, r_3]^T\) as the relative vector of the camera with respect to the celestial center, \(A\) as the apparent diameter of the celestial body, \(D\) as the actual diameter:

\begin{equation}
|\mathbf{r}_{\text{BN}}| = \frac{1}{2} \sin \left(\frac{D}{2A}\right) \tag{7}
\end{equation}

\begin{equation}
\frac{1}{r_3} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = \frac{1}{r_3} \mathbf{e}_{\text{BN}} = \frac{1}{T} \begin{bmatrix} x \\ y \end{bmatrix} \tag{8}
\end{equation}

These equations have been used in multiple instances.\textsuperscript{1,43} The third component of \(\mathbf{r}_{\text{BN}}\) provides the range measurement to the body which can be extracted using the apparent diameter measurements. Hence the definition of \(\mathbf{r}_{\text{BN}}\) which only contains the first two components of \(\mathbf{r}_{\text{BN}}\). The vector components of \(\mathbf{r}_{\text{BN}}\) are expressed relative to the inertial frame assuming inertial attitude knowledge from other instruments. Using the position of the camera on the spacecraft, this provides the measurement value for an orbit determination filter using a circle-finding algorithm.

In the case of the geometric formula, the partials allow to quantify error due to the camera specifications. Indeed, if \(X,Y\) are the pixel sizes (in their respective directions), \(x,y\) are the position on the camera, and \(x_i, y_i, \rho_i\) are the pixel values for the CAD measurements:

\begin{equation}
\mathbf{e}_{\text{BN}} = \frac{r_3}{T} \begin{bmatrix} x \\ y \end{bmatrix} = \frac{r_3}{T} \begin{bmatrix} x_iX \\ y_iY \end{bmatrix} \tag{9}
\end{equation}

\begin{equation}
|\mathbf{r}_{\text{BN}}| = \frac{1}{2} \sin \left(\frac{D}{2A}\right) \tag{10}
\end{equation}

\begin{equation}
= \frac{1}{2} \sin \left(\arctan \left(\frac{\rho}{T}\right)\right) \tag{11}
\end{equation}

\begin{equation}
\frac{1}{r_3} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = \frac{1}{r_3} \mathbf{e}_{\text{BN}} = \frac{1}{T} \begin{bmatrix} x_iX \\ y_iY \end{bmatrix} \tag{12}
\end{equation}

Eq. (9) provides a simple partial with respect to the center...
measurement $c_i = [x_i, y_i]^T$

$$\frac{\partial \rho_{BN}}{\partial c_i} = r_3 \begin{bmatrix} \frac{X}{Y} & 0 & Y \\ 0 & Y & 0 \end{bmatrix}$$  \hspace{1cm} (13)

$$\Rightarrow E[\delta \rho_{BN} \delta \rho_{BN}^T] = r_3^2 \begin{bmatrix} \frac{X}{Y} & 0 & Y \\ 0 & Y & 0 \end{bmatrix} \begin{bmatrix} \delta c_i \delta c_i^T \end{bmatrix} \begin{bmatrix} \frac{X}{Y} & 0 & Y \\ 0 & Y & 0 \end{bmatrix}$$  \hspace{1cm} (14)

The partial for Eq. (10) is:

$$\frac{\partial |\rho_{BN}|}{\partial \rho_i} = \frac{D dx}{2} \sqrt{f^2 + \rho^2 d_z^2} \left( \frac{1}{f^2 + \rho^2 d_z^2} - \frac{1}{\rho^2 d_z^2} \right)$$  \hspace{1cm} (15)

Equation 15 is validated by Monte-Carlo analysis and compared as well to an unscented transform in Figure 4. This shows 10,000 points propagated through Equation 7 using the camera parameters in Table 1 for a range of 18,000 km with Mars offset from the image center by (23, 19) pixels. The pixel standard deviations used are $\sigma_x = 0.5$ and $\sigma_p = 2$.

Figure 4 shows good accordance of the first variations to the Monte Carlos. The image processing methods used here for center and apparent diameter are Hough Circle transforms$^{16}$ 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 5 shows every 30th circles found in the scenario using Hough Circles to fit Mars. Similarly to Figure 3, the x and y axes are pixel numbers in the focal plane, and the colors illustrate the order of the image capture: from dark to light shades. It is immediately seen that the variability of this method is greater than that of the limb extraction. Yet under the assumption of frequent images, when the variations are mostly Gaussian in nature they can be handled through filtering. In practice, the circles are given in a frame centered at the top-right corner of the frame, and the pixel size isn’t directly given. Therefore the computation for the planet direction is given using $X = \frac{\text{SensorSize}}{\text{Resolution}_x}$ and $Y = \frac{\text{SensorSize}}{\text{Resolution}_y}$ in mm/pixel:

$$c_{\rho_{BN}} = \begin{bmatrix} \frac{X}{Y} \cdot \left( x_i - \frac{\text{Resolution}_x}{2} + \frac{1}{2} \right) \\ \frac{Y}{Y} \cdot \left( y_i - \frac{\text{Resolution}_y}{2} + \frac{1}{2} \right) \end{bmatrix}$$  \hspace{1cm} (16)

Where the $\frac{1}{2}$ centers the point on the activated pixel. The normalization by the focal length $f$ functionally brings the pictured spacecraft onto the image plane. This value is then scaled by $|\rho_{BN}|$ computed in Equation 10, and rotated into the desired frames. This is done using the star tracker estimate of $[BN]$ and the known camera frame $[CE]$.

Relative Orbit Determination. Now that two methods have been developed to extract measurements from images, they can be filtered in a relative OD estimator. The measurement noise is extracted from the images and rotated into the proper frame. Figure 6 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 planet size. It should
be noted that although the Hough Circle measurements provide more noise (seen in Figure 6(a)), it is not variable over the orbit and can be handled well by the filter. Figure 6(b) shows the errors from the limb-fitting method. A signal appears in these residuals — which can also be seen in the Hough transformation. 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 slight offset. The errors are, in both cases not necessarily due to the OpNav transforms used, but rather to the image processing method implemented. The filter also overestimates the noise on the measurements by a factor of 10 in order to get the desired results. This is equivalent to modifying the scale factors on the following measurement noises, but provides the filter with noise scaling control. These noise values are selected accordingly:

- The limb-finding algorithm takes as its pixel uncertainty the ratio of the camera width to the number of limb-points found (N), multiplied by 30. The scaling is an empirical value in order to get the correct order of magnitude for the noise:

\[ \sigma_{\text{pix}} = 30 \frac{\text{Resolution}}{N} \]

- The Hough Circle algorithm uses the voting system inherent to the Hough transform. It takes the ratio of the votes accumulated for the circle by the vote threshold, and multiplies it by 0.5:

\[ \sigma_{\text{pix}} = \frac{1}{2} \frac{\text{votes}}{\text{voteThresh}} \]

Both the methods described provided relative position measurements for the orbit determination filter. This filter which estimates spacecraft position and velocity in the inertial frame is implemented as a square-root unscented Kalman filter. This filter is used both for alongside star tracker measurements for inertial attitude, and in conjunction with heading determination filters using the planet’s centroid for relative attitude. The filter state is:

\[ X = \begin{bmatrix} N \vec{r}_{BN}^N \\ \dot{N} \vec{r}_{BN}^N \end{bmatrix} \]  

(18)

Where \( N \vec{r}_{BN}^N \) is the spacecraft position relative to the celestial body (Mars). The ‘dot’ represents a derivative as seen by the inertial frame. This keeps the estimation process minimal, though other states could be added if on board applications allow.

\[ \dot{X} = F(X) = \begin{bmatrix} \dot{\vec{r}}_{BN} \\ \frac{\ddot{\vec{r}}_{BN}}{|\vec{r}_{BN}|} \end{bmatrix} = \begin{bmatrix} \dot{\vec{r}}_{BN} \\ -\frac{\dot{\vec{r}}_{BN}}{|\vec{r}_{BN}|^2} \vec{r}_{BN} \end{bmatrix} \]  

(19)

The dynamics of the filter are given in Equation (19). The state propagation is done using an RK4 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 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:

\[ G(X) = N_r R_N \]  

(20)

This is done to simplify the upkeep and modularity of the filter for OpNav, but also to be able to use different types of measurements for one same OD estimate.

**Coupled 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: 1 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 7. Relative Errors — NIH-SVD](image)

The **Hough Circle** method performs well overall, as seen in Figures 8 and 10, given the simplicity of the algorithm and the sensitivity of the position computation. The limb extraction paired with NIH-SVD outperform it, given that the **Hough Circle** covariances range between 200 and 500km (\(~2.2\%\) error on position and 0.06km/s (\(~9\%) on velocity. The state estimates have errors in percentages below 0.9\% on position and below 2\% on velocity. These errors oscillate depending on the lighting conditions, 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 are driven by the elliptical nature of the orbit (true velocity decreases and relative distance increases).

![Figure 8. Relative Errors — Hough Circles](image)
finding algorithm. Indeed, sub-pixel edge localization algorithm using Zernike moments would allow for better results with the same images.

**Outlier Detection and Mitigation.** 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. This is notably used to assess the feasibility of missions utilizing autonomous or ground-based OpNav in order to simulate hardware and software limitations. When testing and validating autonomous systems, testing for the “off-nominal” case is crucial to understanding the limits of the performances and mission feasibility. These situations can also arise from sensitivity analysis through Monte-Carlos allows the comparison of expected algorithm performance against mission requirements.

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.

The simulation can generate multiple errors, unexpected objects or artifacts, and other corruptions in the images. A way to test the methods is to bring in a distance object in the background. Although the Sun is currently used — no camera damage is simulated, and the Sun just appears as a cluster of white pixels — ongoing work is adding in noise, cosmic rays, and other perturbations. Therefore the Sun saturates a set of pixels but doesn’t overflow adjacent pixels.

<table>
<thead>
<tr>
<th>Table 6. Fault Initial States</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_{BN}$</td>
</tr>
<tr>
<td>$\omega_{BN}$ [rad/s]</td>
</tr>
<tr>
<td>Orbital Elements</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Table 6 shows the initial conditions for the fault simulation. No dynamics are changed, and the only difference in the flight software algorithms is that the pointing is
forced on the planet (knowing that it is the zero-base of the Spice reference frame). Although this breaks the coupling between attitude or orbit determination that was used in previous simulations, it generates specific conditions of interest in this section.

The difference in the two image processing methods was seen in Figures 11. Although a coarse perturbation, having an unexpected object come through the camera field-of-view could be representative of dust specs, or camera artifacts. These corruptions aim to improve on the global robustness of OpNav algorithms.

The limb processing module has room for improvement in order to protect against faulty limbs. If the lesser performance of Hough Circles can be seen in Figure 10, it at least provides some robustness. This framework not only provides comparative results between two identical methods, but also runs the two methods side-by-side in one simulation. Although the Hough Circle method is developed with the same level of protection against errors, this section attempts to heighten the ability of the simulation to highlight the differences between these methods.

The benefit of this tool beyond that of comparison, is to harness the strengths of each implementation in order to reject faults. In this case, a comparative module for image fault detection can reject the measurements from the limb finding method when not in accordance with the circle-finding algorithm. Hough Circles, in their simplicity provide assurance of a certain type of feature being processed, but remain noisy measurements. Limbs on the other hand provide much more refined measurements, but can be more susceptible to artifacts and even lighting conditions.

Figure 12 shows the data after having been processed by their respective algorithms. These two algorithms were run side-by-side and can be compared in several different ways to create a more robust autonomous algorithm altogether.

**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 doesn’t perform to the same level as current state of the art algorithms, it provides a robust alternative with a simple implementation. The difference between state-of-the-art algorithms and the new Hough Circles method is exemplified throughout the paper, notably given the fault detection section. Finally the entire study is done with a underlying pointing coupling with the orbit determination. Doing simultaneous attitude 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 show valid results for autonomous navigation around Mars. Future work includes more in-depth developments in fault detection, Monte-Carlo analysis for sensitivity, and new method implementations.

The authors would like to acknowledge the continu-

Figure 11. Image Processing in case of faults

Figure 12. Impact of faults on measurements in C

References.


