AAS 22-146

CRATER NAVIGATION AND TIMING FOR AUTONOMOUS LUNAR ORBITAL OPERATIONS IN SMALL SATELLITES

Z R. McLaughlin*, Rachael E. Gold*, Sofia G. Catalan*, Rahul Moghe*, Brandon A. Jones†, and Renato Zanetti†

Efforts at NASA and in the commercial space sector seek to extend space operations into cislunar space, lunar orbit, and down to the Moon's surface. Mission concepts require the extension of infrastructure and support services into these new regimes to enable comparable autonomous and safe operations seen in the near-Earth environment. This paper presents research to develop a Crater Navigation and Timing (CNT) technology capable of running in a 3U CubeSat that is consistent with the operating conditions of the lunar environment. The CNT approach requires three algorithms running on-board the spacecraft: detection of craters in an optical image, identification of those surface features, and fusion of these observations with a predicted trajectory to produce a position, navigation, and timing (PNT) solution. This paper will demonstrate the fusion of these algorithms and show the PNT performance of the software under small satellite flight-like constraints.

INTRODUCTION

As the scope of space missions beyond Earth orbit widens, cislunar space has become an area of increased focus for NASA, the DoD, and many other government and corporate agencies. The potential for further technological advancement, planetary science opportunities, and space exploration is high, but there is a clear need for new technologies to enable an autonomous set of missions to achieve these goals.[1] Among some of the most important capabilities necessary to enable future missions of this type is the ability to navigate with accuracy and precision in lunar orbit.

Navigation in the Moon's orbital environment poses a challenge different to that of navigation in terrestrial orbits. Because of a lack of GNSS systems and the increased cost of ground-based communication to lunar assets, there is a far greater cost in maintaining position and timing knowledge of a spacecraft orbiting the Moon rather than the Earth. Due to limited resources and communication opportunities, the ability for such a spacecraft to gather measurement data to update its knowledge would greatly increase the feasibility of deploying and managing more assets while potentially enabling a network that could share this navigation and timing data with other spacecraft. Crater Navigation and Timing (CNT) development is dedicated to the advancement of such lunar technologies by achieving more precise navigation and timing solutions in lunar orbit using in-situ measurements to improve and maintain a priori state knowledge.

*Graduate Research Assistant, Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, Austin, TX 78712

†Assistant Professor, Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, Austin, TX 78712.

One of the most readily available types of measurement data in the lunar environment are the craters that populate the Moon's surface. Terrain Relative Navigation (TRN) is a form of navigation that uses the natural features of a body to provide measurements through identification and catalog matching. This may be done by comparing information gathered from optical data to a catalog of a body's features. For this work, the craters on the Moon's surface are used in conjunction with an optical camera and a neural network to provide estimates of visual craters' locations. These located crater's pixel values in an image may then be matched to a catalog of the Moon's surface features to provide a full measurement model and observation data for the spacecraft.

In the past, similar efforts to that of this project have been undertaken to utilize TRN-based approaches in lunar orbit.[2] Until recently, these efforts have utilized classical image processing techniques such as fast Fourier transforms (FFTs), shadow modeling, and pixel intensity thresholding in order to produce a centroid location. As technology has continued to advance in the field of machine learning (ML), however, numerous advantages of ML-baed detection algorithms have been identified, such as further robustness to crater irregularity, higher accuracy in a variety of lighting conditions, and performance without geometric constraints. Robustness in cases like this have led to a recent proliferation in other ML-based lunar navigation work,[3,4] which may be more flexible when considering the varying surface geometry and lighting conditions common to the craters on the surface.

The CNT problem may be roughly broken down into three primary components; crater centroid detection, catalog identification, and position and time estimation. The goal of this work is to produce a stable final product which is able to produce a navigation solution with a root mean square (RMS) position error of under 100 meters and an error in time under 100 milliseconds. Current results provided in this paper show that each component element of the project is able to perform with consistent accuracy for near circular and near equatorial low-lunar orbit test cases (those having Keplerian orbital elements of inclination under 20 degrees and 0.001 eccentricity). This paper also presents additional tests to indicate tentative success in integration between the different component parts, with current tests showing successful navigation state estimation using the machine learning detector for up to 3 orbit periods while meeting the position threshold accuracy. Time bias estimation algorithms have also been added to the software and show that the initial target of time estimation error within 100 milliseconds is possible, but further work here to compensate for more realistic clock bias models is still underway.

CRATER DETECTION

Processing the terrain images taken by an on-board camera enables the subsequent state estimation in this work. The obvious task of the crater detection method is to determine where craters are in an image, however, the wide variety of what craters can look like presents a challenge to autonomous processing. Lighting conditions, along with the crater size, shape, depth, and surrounding terrain, greatly affect how a crater appears in an image, which is why the detection process requires computer vision methods that are more complex than simple edge and circle detection.

Mask R-CNN Detector

The crater detection method used in this paper expands on previous work that describes the use of a Mask R-CNN model.[5] While the prototyping and development of trained Mask R-CNN model was done in the PyTorch framework[6] and Python, the CNT software was developed in C++ with an object-oriented programming structure. To integrate the crater detection model into C++, the

Open Neural Network Exchange (ONNX) format was used, along with their open sourced tools.[7] ONNX enables the PyTorch-based model to be converted such that the C++ CNT code can run the detector in the loop for further development and integrated testing. These machine learning-based capabilities enable the crater detection method to handle the variability in lunar craters to provide detections with an associated confidence value. The terrain images are passed through the detector and each crater detection is tagged with a confidence value, which enables sorting out low confidence craters with the goal of removing false detections. Following an ellipse fitting method for each detected crater, a conversion process is done to compute the horizontal and vertical bearing angles of the crater centroid relative to the camera boresight via

$$\alpha = \arctan\left(\frac{x_c - 0.5N}{f}\right) \text{and} \tag{1}$$

$$\beta = \arctan\left(\frac{y_c - 0.5N}{f}\right), \tag{2}$$

where (x_c, y_c) are the crater centroid pixel coordinates, N represents the image pixel length, f is the camera focal length, and (α, β) are the horizontal and vertical angles respectively for an input square image of the lunar terrain.

The integrated detector in this paper improves upon previous work[5] with changes to enable more accurate centroid estimation for craters that are partially in the field of view. This improvement adds more crater detections to be processed by the identification step, which requires that the computed centroid is within a specified threshold of the true catalogued crater. With this partial crater centroiding capability, the estimation filter gains additional measurements that can be processed in the measurement updates.

Image Generation and Datasets

Testing the crater detection method developed in this work requires lunar terrain images that are representative of those taken by an orbiting spacecraft. Images taken by previous and ongoing missions provide the data needed to produce simulated imagery, or in some cases, like for Apollo and the Lunar Reconnaissance Orbiter, direct imagery from missions provide real images that can be processed through the crater detection software. For algorithm development and evaluation, the use of simulated imagery provides several advantages: (1) perfect knowledge of the spacecraft trajectory and dynamics models, (2) configurable camera model, (3) idealized lighting conditions, (4) known true crater locations, and (5) automated scripting for testing.

While testing individual images against crater detection methods provides insight on the crater centroiding accuracy, the subsequent processes including the crater identification and spacecraft state and time estimation can only be tested with a continuous set of images that are representative of a spacecraft in orbit. Given a user-specified dynamics model, the generation of an input trajectory can simply use a two-body orbit around the Moon or with higher fidelity models and numeric integration of the equations of motion. Then, setting a camera model enables the generation of terrain image sets that can be used to test the entire CNT workflow.

For the development of the crater detection method used in this paper and integrated CNT testing, the primary data sources for generating images are the LROC Global Morphologic Maps[8] and the Robbins lunar crater database.[9] With software tools developed in previous work, the LROC maps and the crater database enable an automated capability to generate simulated datasets with continuous trajectories.[5] As the image set metadata contains truth information for where the craters are

located, the centroiding accuracy of the crater detection method and the true spacecraft state can be compared against the computed values through the CNT process. Scripts using simulated imagery are responsible for numerically validating the developed algorithms so that all inputs to the image and trajectory generation software may be specified by a user.

Figure 1 shows an example set of images with highlighted crater detections from the LROC-based simulated imagery. In future work, image generation through tools like PANGU[10] and Blender[11] will be explored to enable modeling different lighting conditions, terrain shadows, and more accurate camera models along with the use of Apollo and LRO mission data to add further testing capabilities for the CNT framework.

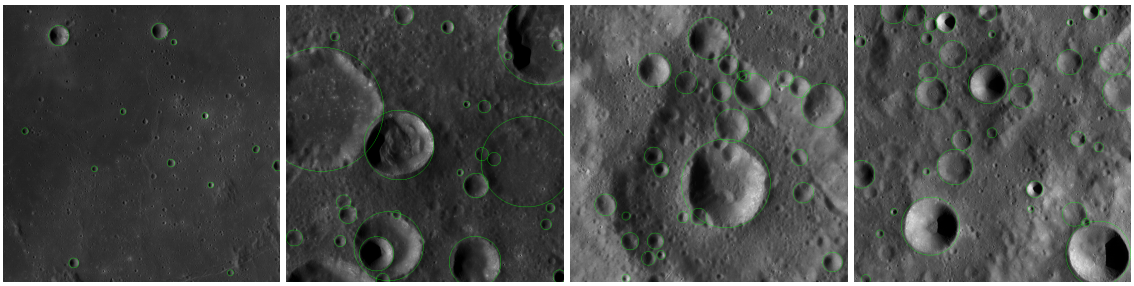


Figure 1: Simulated images from LROC Global Maps with crater detections

TIME BIAS ESTIMATION

Estimation of the time onboard a potential lunar asset is crucial in order to ensure the correct use of ephemeris-based data to perform navigation filer time updates and to ensure that incoming biased timestamps are not used as-is to perform updates to the navigation state. The CNT software is currently capable of receiving an a priori PDF from a simulated ground station to estimate the bias time. This may be done every certain number of measurements, or at any other time, so long as the input states are thought to be at the same time (though the navigation time is in fact biased). The time bias is

$$\Delta t = t - t^*, \tag{3}$$

where t is the true time and t^* is the time measured by the clock on the spacecraft. This work seeks to estimate Δt. In order to facilitate this process as it is being further developed, simplifying assumptions are made by the CNT algorithms to allow for a smoother time estimation process:

- There is no error in the time tag of the ground-based ephemeris.
- The clock bias is estimated using one navigation solution rather than a batch.
- The potential uncertainty in the ground solution is not considered.
- The bias estimator considers the reference state to be true.
- The navigation covariance does not change over the bias period when propagating.
- The algorithm does not consider the velocity covariance in the navigation solution.
- The algorithm assumes that:

$$\frac{\partial \boldsymbol{r}\left(t^* + \Delta t\right)}{\partial \Delta t} \approx \boldsymbol{v}\left(t^* + \Delta t\right) + \left(\boldsymbol{a}\left(t^* + \Delta t\right)\right) \Delta t \tag{4}$$

In order to actually perform the bias estimation, the CNT software solves the nonlinear least squares problem

$$\widehat{\Delta t} = \arg\min_{\Delta t} \left(\boldsymbol{r}(t) - \hat{\boldsymbol{r}}\left(t^* + \Delta t\right)\right)^T P^{-1} \left(\boldsymbol{r}(t) - \hat{\boldsymbol{r}}\left(t^* + \Delta t\right)\right). \tag{5}$$

This illustrates that the algorithm attempts to converge on a Δt which minimizes the difference between the estimated filter solution, $\hat{\boldsymbol{r}}\left(t^* + \Delta t\right)$, and the ground epoch solution, $\boldsymbol{r}(t)$.

To independently verify this proof of concept, Monte Carlo trials was performed to test against the accuracy limit of 100 ms. These results were gathered using a simple two-body propagation scheme, though it is important to note that the propagator fidelity may be customized within the CNT software and that further results have been gathered using a higher fidelity propagator with gravity gradient perturbations.

For the test results included here, two different biases of one and ten seconds were used to set the true and biased states apart. The scripts utilized a navigation solution at time t^* along with the true bias of one second to create 10,000 randomized realizations of the true state at the true time. Each of these true states were used in the trial to produce the statistical performance of the time bias estimation algorithm.

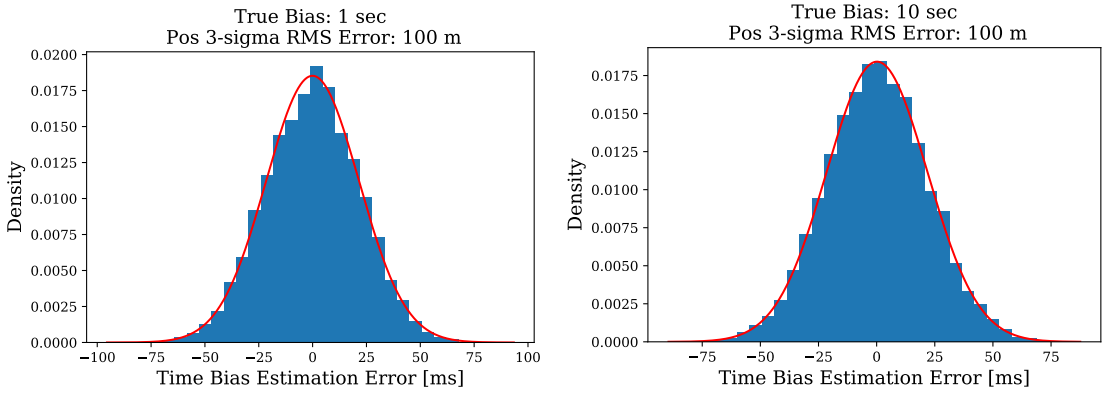


Figure 2: Monte Carlo time bias results

Figure 2 shows that the time bias estimator is able to successfully track the difference in the biased and real times in order to apply the differential to the estimated state. Additionally, these results show that the algorithm is capable of doing so consistently with an error less than 100 ms.

Although the time bias estimation software has been fully integrated with the simulation, detection, and filtering capabilities of the CNT software, at this point in time the realism of the clock bias simulation could be improved significantly. The time bias is currently simulated as a constant offset value to the true timestamps, so there is no walk in each timestamp as continued measurements are received by the filter.

NAVIGATION FILTER

Estimating the state of a spacecraft in low lunar orbit is the principle goal of the CNT project. Both the time bias estimation capabilities and the crater identification and matching algorithms are meant to bolster and enable the software's ability to maintain a robust navigation solution. The navigation filter implemented in the CNT software utilizes these elements in conjunction with data

provided by the crater detector in order to update and propagate the a priori state's probability distribution function (PDF). A slightly modified extended Kalman filter (EKF) is used in this software in order to ensure on-board tractability when implemented on flight hardware compatible with a small satellite. The position estimator part of the CNT software is compatible with a variety of different propagators, with a range available from simple two-body models to those including solar radiation, gravity gradient perturbations, and drag models to those performing ephemeris-based propagation. Any change to the propagator choice is reflected across the other CNT applications, such as the time bias estimation software.

The navigation algorithm works in two discrete parts, the time update and the measurement update, with a few important steps in between to ensure the timely processing and matching of the detections and craters. This section outlines each of these components in order to illustrate how they fit together to produce an updated PDF. The time update for the CNT object is performed before crater matching and identification; this update is responsible for receiving the timestamp of the incoming image measurement, applying the estimated time bias to that image to attempt to correct the onboard clock bias, and propagating forward to the estimated image time. After this point, further processing is required in order to prepare for the measurement update. First, a local crater catalog must be generated based off of the estimated position and attitude of the spacecraft, which is assumed to be provided from an external source, such as an IMU on board. After crater identification and matching, a measurement update is performed. The measurement model used in the filter classifies bearing angles, denoted as α and β as the observation vector. This information is calculated using the pixel values of the estimated centroids and the MCI position of the crater centroid calculated from the corresponding latitude and longitude of the crater determined through matching. The measurement update itself is a modified version of a classical EKF update, with an underweighting factor to account for nonlinearities as well as an iterative update which uses every matched crater to accumulate a proposed change to the a priori state before finally summing and applying those changes to the a priori mean and covariance. The latter parts of this section describes each of these three key processes; the catalog creation and crater matching, the measurement model used in the filter, and the measurement update.

Local Catalog Creation and Crater Matching

The CNT software is built to receive a measurement in the form of an image. This means that the only immediately available information is the image timestamp and a collection of pixel centroids which may be retrieved from the Mask R-CNN detector. This information by itself is of no help to the spacecraft, meaning that further knowledge is required to utilize those detections towards a state update. This information can be retrieved by use of a surface feature catalog, in this case the Robbins lunar crater catalog. This database, pre-truncated for crater size based on camera field of view and orbit regime, is used to create a local catalog at each time step by projecting the spacecraft camera's field of view pointing downward towards the surface according to the spacecraft's body frame. For this work, the body frame is the North-East-Down down frame (though this may be otherwise specified as an input to the CNT software by the user).

After a local catalog is created, the CNT algorithm utilizes the Munkres or Hungarian Matching algorithm[12,13] in order to identify craters based on entries in the crater catalog. In the Munkres algorithm, assignment is determined based on a cost matrix, $\boldsymbol{C}$. The entries in the cost matrix are filled with the 2-norm of the distances, $\boldsymbol{d}$, in pixel space, between the craters detected by the

detector, $\boldsymbol{r}_{detected}$, and the craters in the catalog, $\boldsymbol{r}_{catalog}$ via:

$$\boldsymbol{C}(i,j) = \|\boldsymbol{d}(i,j)\|_2 = \|\boldsymbol{r}_{detected}(i) - \boldsymbol{r}_{catalog}(j)\|_2 \tag{6}$$

where $i = 1:n$, $j = 1:m$. Assignment is then determined by finding the minimum of:

$$\hat{\pi} = \underset{\pi \in \Pi}{\arg\min} \left[\sum_{i=1}^{n} \min\left(c, \boldsymbol{C}(i, \pi(i))\right) \right] \tag{7}$$

where Π is the set of all assignments of detected to cataloged craters, π, and c is the cutoff value for assignment. For the CNT algorithm, the cutoff value is chosen to be the maximum value of $\boldsymbol{C}(i,j)$ shown in Eq. 6, which mitigates the risk of false identification of craters.

Measurement Model

The measurement model utilized in the CNT framework relies on bearing angles for observations. These bearing angles, denoted as α and β, represent the location of the crater centroid compared to the center of the image in horizontal and vertical field of view (FOV) space, respectively. Figure 3 shows the localized camera attitude reference frame and how the bearing angles are represented in it. The bearing angles can be calculated directly using the vector from the camera to the crater centroid

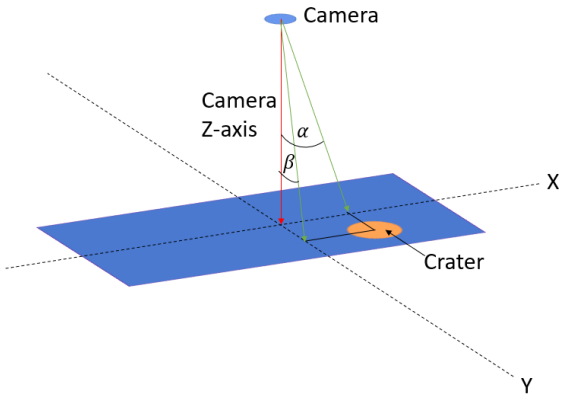


Figure 3: Camera Axes and Bearing Angles

represented in the camera frame as shown in Eq. 8. In order to perform this calculation, the transformation matrix $\boldsymbol{T}_{CAM}^{MCI}$ denoting the transformation from the Moon-centered Inertial reference frame (MCI) to the camera frame (CAM) must be calculated, shown in Eq. 9.

$$\left[\boldsymbol{r}_{sf}^{CAM}\right]_{CAM} = [x\ y\ z\ ] = \boldsymbol{T}_{CAM}^{MCI}\left([\boldsymbol{r}_{sf}]_{MCI} - [\boldsymbol{r}_{orb}]_{MCI}\right) \tag{8}$$

$$\boldsymbol{T}_{CAM}^{MCI} = \boldsymbol{T}_{CAM}^{S}\boldsymbol{T}_{S}^{B}\boldsymbol{T}_{B}^{MCI} \tag{9}$$

The B and S subscripts in Eq. 9 represent the body frame and the structure frame, respectively. By taking the x, y, and z components of the $\left[\boldsymbol{r}_{sf}^{CAM}\right]_{CAM}$ vector from Eq. 8. The bearing angles are

$$\alpha^* = \arctan\left(\frac{x}{z}\right) \text{ and} \tag{10}$$

$$\beta^* = \arctan\left(\frac{y}{z}\right). \tag{11}$$

The partial derivatives of the bearing angles in the MCI frame can be defined as

$$\frac{\partial \alpha^*}{\partial [\boldsymbol{r}_{orb}]_{MCI}} = \frac{\partial \alpha^*}{\partial [\boldsymbol{r}_{sf}]_{MCI}} \frac{\partial [\boldsymbol{r}_{sf}]_{MCI}}{\partial [\boldsymbol{r}_{orb}]_{MCI}} = \frac{\partial \alpha^*}{\partial [\boldsymbol{r}_{sf}]_{MCI}} \boldsymbol{T}_{CAM}^{MCI} \left[\boldsymbol{r}_{sf}^{CAM}\right]_{CAM} \text{ and} \tag{12}$$

$$\frac{\partial \beta^*}{\partial [\boldsymbol{r}_{orb}]_{MCI}} = \frac{\partial \beta^*}{\partial [\boldsymbol{r}_{sf}]_{MCI}} \frac{\partial [\boldsymbol{r}_{sf}]_{MCI}}{\partial [\boldsymbol{r}_{orb}]_{MCI}} = \frac{\partial \beta^*}{\partial [\boldsymbol{r}_{sf}]_{MCI}} \boldsymbol{T}_{CAM}^{MCI}, \tag{13}$$

where the partials with respect to the surface features in the MCI may be first calculated as

$$\frac{\partial \alpha^*}{\partial [\boldsymbol{r}_{sf}]_{MCI}} = \begin{bmatrix} -\dfrac{z}{z^2+x^2} & 0 & \dfrac{x}{z^2+x^2} \end{bmatrix} \text{ and} \tag{14}$$

$$\frac{\partial \beta^*}{\partial [\boldsymbol{r}_{sf}]_{MCI}} = \begin{bmatrix} 0 & -\dfrac{z}{z^2+y^2} & \dfrac{y}{z^2+y^2} \end{bmatrix} \tag{15}$$

in order to more feasibly compute the full partial derivatives with respect to the orbital state. The partial derivatives of the bearing angles with respect to the orbital velocity of the satellite are simply zero vectors for both α^* and β^*.[14]

Measurement Update

Now that the crater identification and matching algorithm and measurement model have both been defined, the measurement update may be dictated clearly. As previously mentioned, an EKF is the filter responsible for performing measurement updates in the software. This EKF has two primary modifications to the traditional algorithm; one is the way in which the filter uses the calculated residuals to produce a deviation to alter the state, and the other in the measurement underweighting included in the algorithm.

The measurement underweighting used in this filter closely follows the presentation of underweighting methods described by NASA's *Navigation Filter Best Practices* handbook.[15] In particular, the software contains options describing the simplest underweighting procedure — scaling the measurement error covariance matrix — or underweighting following their presentation of Lear's method, which is currently in use in the CNT filter algorithm, to produce an altered Kalman gain $\mathbf{K}_k$,

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left((1+\gamma)\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k\right)^{-1}, \tag{16}$$

which is scaled by the factor γ. Here, $\mathbf{P}_k^-$ denotes the a priori covariance, $\mathbf{H}_k$ is the Jacobian matrix of the measurement model, and $\mathbf{R}_k$ is the measurement covariance matrix. Underweighting was included in the filter due to the linearization error incurred by the adoption of the EKF in solving nonlinear problems; it is possible that the a posteriori uncertainty as computed by the filter does not represent the truth to an accurate enough extent such that the estimated covariance is inconsistent with the errors in the state. This can be a valid area of concern for the CNT problem because of the large fluctuation in the number of detections per image; a period of sparse detections can cause the a priori uncertainty to grow quite large. In such a case, a measurement with many accurately

identified caters may actually reduce the size of the covariance too rapidly, negatively affecting filter performance. With the inclusion of the underweighting scheme, it is possible to mitigate this issue.

One further change from a standard EKF is that the measurement update processes a batch of measurements which should all be able to adjust the a priori mean with equal weight. The CNT filter does this by accumulating a deviation in the mean for each measurement loop before modifying the a priori. In other words, an individual a posteriori mean is not calculated and then updated again for and by each centroid measurement; rather, the difference that each measurement would apply to the mean is accumulated across each measurement while the covariance is adjusted during each individual update. After all updates have been processed, the final deviation is applied to the state, thus completing the generation of the a posteriori distribution.

The filter's process noise covariance, $\mathbf{Q}$, is generated using the linear process noise model,[16] as

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{4}\Delta t^4 \boldsymbol{q} * I_3 & \frac{1}{2}\Delta t^3 \boldsymbol{q} * I_3 \\ \frac{1}{2}\Delta t^3 \boldsymbol{q} * I_3 & \Delta t^2 \boldsymbol{q} * I_3 \end{bmatrix}, \tag{17}$$

where $\boldsymbol{q} = [q_1 \ q_2 \ q_3]$ is a vector of constants which can be tuned as needed, Δt is the change in time from the previous measurement to the current measurement, and I_3 is a three by three identity matrix. The results presented in this paper use process noise values of $q_1 = q_2 = q_3 = 10^{-15}$.

Finally, the filter also makes use of measurement editing to ensure that only measurements that result in a distribution statistically consistent with a chi-squared distribution are used in the state estimate. This is done by checking the squared Mahalanobis distance,

$$m_r^2 = \Delta \boldsymbol{z}_k^T (\mathbf{P}_k^+)^{-1} \Delta \boldsymbol{z}_k, \tag{18}$$

where $\Delta \boldsymbol{z}_k$ is the pre-fit residual and $\mathbf{P}_k^+$ is the a posteriori covariance. When m_r is less than the inverse of the cumulative distribution function of the χ^2 distribution of the order of the degrees of freedom of the observations.[15] Since the bearing angles α and β are the measurements, the order of the chi-squared distribution is two and the boundary for the m_r^2 is 5.9915 for 95 percent confidence.

RESULTS

This section details the results produced by the CNT algorithms, both with and without the machine learning detector in the loop. All of the CNT software meant for eventual flight-compatible implementation is implemented in C++ and C, while the simulation utilities are primarily housed in Python scripts. The simulation architecture developed for this work also allows for a variety of propagator fidelities to generate true states. This propagator selection wraps the same utilities used for propagation on in CNT algorithms. These scripts maintain a data pipeline which is responsible for handling a given test case and then generating all of the required information to run the CNT software. This information consists of a true trajectory file for comparison and a crater centroid detections file which is written for several different sub-cases; simulated detections with and without measurement noise and with and without a lunar dark side effect for each of the noise cases, where those craters which are not illuminated are not "seen" by the detector. In addition to the simulated detections file, a file is also generated from the Mask R-CNN detector using ONNX with the same trajectory and attitude information used to generate the simulated trajectory. This information is used in concert with the LRO maps to produce an analog to the simulated detections file which uses previously discussed simulated lunar surface imagery to truly test the CNT algorithms and how well they perform.

Test Cases and Data Generation

Table 1: Test cases

Cases	a (km)	e	i (rad)	Ω (rad)	ω (rad)	ν (rad)
0	1837.4	0.0	0.0	0.0	0.0	0.0
1	1837.4	0.001	0.0	0.0	0.0	0.0
2	1837.4	0.0	0.3	0.0	0.0	0.0
3	1837.4	0.001	0.3	0.0	0.0	0.0

This initial publication considers four different test cases. Case zero is an equatorial and circular orbit, case one is an eccentric and equatorial orbit, case two is a circular and inclined orbit, and case three is an eccentric and inclined orbit. Between all of these, the discrepancies between performance between orbit types can be more easily identified. For example, due to the LRO maps projection from a 3D surface to a 2D plane, more distortion is present for images further from the equator. Including slightly inclined cases in the algorithm is a way to test responses to this issue and to see how the filter may respond to image distortion, which would create a mismatch between the pixel location of the centroid and the corresponding latitude and longitude of the same crater on the surface.

Simulating detections rather than using the neural network alone provides a way to debug the filter without worrying about measurement errors and inconsistencies. Because noise can be added at will, the filter may be tested across a variety of different scenarios to better understand its performance. In the simulated detections case, imagery is completely bypassed and the true crater locations from the Robbins catalog are used and edited to create a set of false measurements based on the true trajectory of the simulated spacecraft. The detections are generated by first finding the field of view (FOV) given the position of the satellite assuming a North-East-Down attitude. Equation 19 calculates the FOV,

$$FOV = 2\arctan\left(\frac{L}{2h}\right), \tag{19}$$

where h is the altitude in relation to the Moon and L is the length of the edge of the FOV in kilometers. Note here that the FOV of a camera onboard an actual spacecraft would likely be fixed — this calculation of the FOV is an artifact of simulation constraints. The reason for this is to be compliant with an initial image dataset from LROC wherein each image covered 3 degrees latitude and 3 degrees longitude in either direction. Because the altitude of the spacecraft is not fixed, yet the image size is, a variable field of view is simulated in order to ensure compatibility with the image data. Equation 19 currently assumes a square, flat FOV. Future work will include algorithms to combat the distortion from a 3D surface to a 2D camera projection, but this has not yet been implemented. After the FOV is calculated, it is projected into the camera frame of the satellite via

$$u = v = h\tan\left(\frac{FOV}{2}\right), \tag{20}$$

where u and v represent the distance in the planar directions in the Camera frame. Using u, v, and h, the boundaries for the FOV can be transformed into lunar latitude, longitude, and altitude coordinates. At this point, the craters in the Robbins catalog that lie within the boundaries are modified through the addition of Gaussian white noise to simulate the expected noise from the sensor and crater detection process.

Simulated Detections Results

All four cases are tested with the simulated detections which are shown in Table 1. All cases have detections generated with both ideal lighting, where no craters on the surface are shadowed, and with the simulation accounting for the shadowed area of the Moon, considered to be between between true anomalies of $\pi/2$ and $3\pi/2$. Additionally, the true position and the filtered position are propagated with gravity perturbations calculated using a spherical harmonics model from Lunar Prospector. The ephemerides of the Moon are provided by the JPL DE 421 file for the truth and DE 405 file in the filter propagator.[17,18]

The filter has been tested for consistency with 100 Monte Carlo runs. The Monte Carlo runs are generated by keeping the initial a priori in the filter constant and randomly sampling that same a priori distribution to find an initial state for the the true trajectory. The true trajectory is then used to generate a new set of simulated detections. The MCI state error results for the Monte Carlo runs for Case 3 are shown in Fig. 4 under ideal lighting and Fig. 5 with the dark side of the Moon simulated. In these plots, the two different 3σ bounds are generated from the mean 3σ bound output from the filter and also three times the standard deviation calculated from the 100 Monte Carlo runs. The root mean square error (RMS) for the position for a single run of all cases is shown in Table 2. In each case, the position accuracy target of 100 meters is met consistently for a case with no gaps in measurements caused by lack of surface lighting. For the cases which simulate the spacecraft in shadow, the state error occasionally peaks over 100 meters in some individual Monte Carlo runs before quickly converging towards zero as new measurements become available.

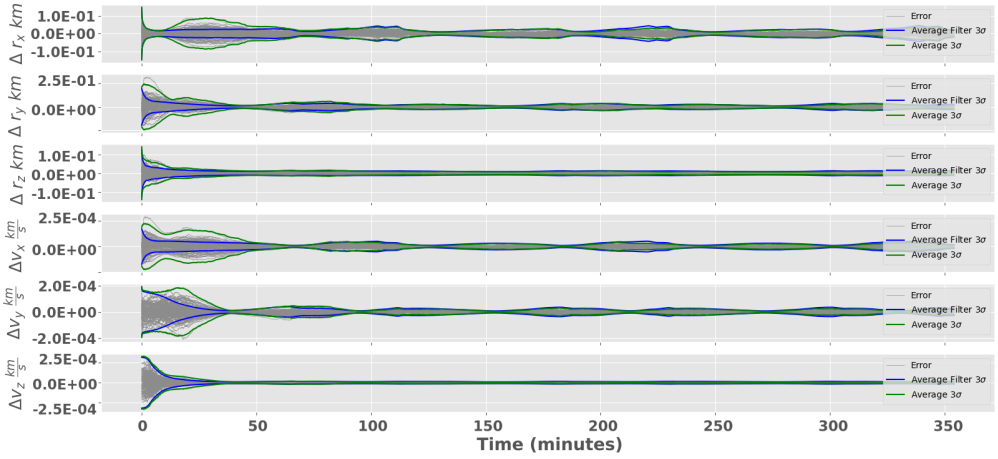


Figure 4: Monte Carlo results for Case 3 with ideal lighting

Time Bias

Test cases 0 and 3, representing the circular and equatorial and the eccentric and inclined cases, respectively, have been used to generate results in the presence of a clock bias. The CNT software models this bias by applying an offset to the incoming image timestamps according to a bias value and a standard deviation that may be altered by the user. Because the software cannot be directly sensitive to random perturbations in the bias outside of the use of process noise in the filter, the

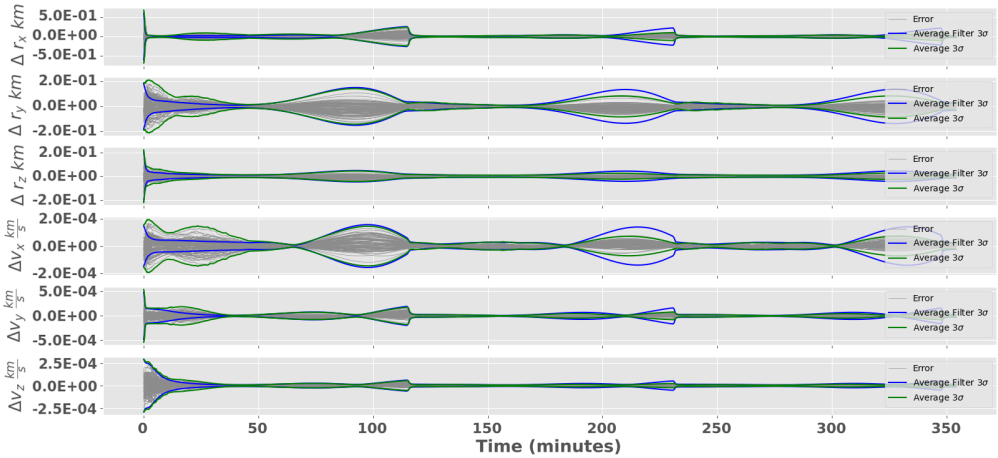


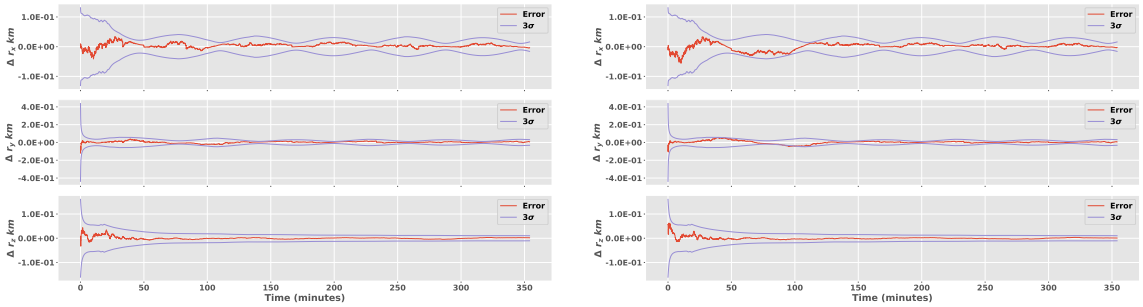
Figure 5: Monte Carlo results for Case 3 with simulated shadow

Table 2: Root mean square error for one Monte Carlo trial of all simulated cases

Cases	RMS_x (km)	RMS_y (km)	RMS_z (km)	RMS_{3D} (km)
Ideal 0	0.011	0.017	0.004	0.021
Ideal 1	0.005	0.005	0.005	0.009
Ideal 2	0.010	0.010	0.007	0.016
Ideal 3	0.015	0.011	0.005	0.020
Dark Side 0	0.044	0.041	0.009	0.061
Dark Side 1	0.032	0.045	0.040	0.069
Dark Side 2	0.078	0.076	0.050	0.120
Dark Side 3	0.079	0.078	0.024	0.114

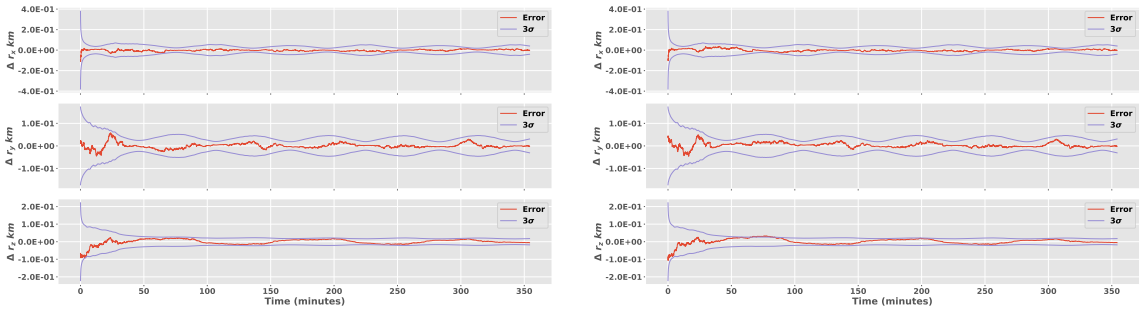
time bias estimation algorithm has only been tested in the loop with a deterministic bias or with very small (on the order of 1E-3 seconds) standard deviation values. These tests show that with process noise as earlier described and with the time bias estimation software in the loop, the CNT software may estimate a constant or near-constant on board clock bias while providing a position estimate with under 100 meters of error. These results also show that a timing solution of under 100 milliseconds of accuracy is within reach, but that it cannot be guaranteed for each individual simulation at this time.

Figures 6 and 7 show the error in the position estimate over time for cases 0 and 3, respectively. For all of the time bias test cases, a 5 detection cap was used for each image measurement to ensure similarity with the Mask R-CNN detector outputs. Each of these two scenarios has two different bias cases; one with a simulated clock bias of 2 seconds and another that has a clock bias of 20 seconds. An a priori update from the ground is provided every 1000 seconds in this proof of concept demonstration. The software models the ground reference state to be accurate to 1E-5 kilometers in position and 1E-8 kilometers per second in velocity to the true spacecraft state. At this time, the CNT software does not utilize covariance information along with the ground update. These tests show that the performance of the CNT software with a constant clock bias is within the goal of a position error less than 100 meters.



(a) 2 second bias, ground update every 1000 seconds (b) 20 second bias, ground update every 1000 seconds

Figure 6: Position state errors with clock bias estimation: case 0



(a) 2 second bias, ground update every 1000 seconds (b) 20 second bias, ground update every 1000 seconds

Figure 7: Position state errors with clock bias estimation: case 3

Table 3 shows the root mean square error for the estimated position of the spacecraft over time for all four runs shown in Figs. 6 and 7. The results here show an acceptably low RMS compared to the results presented in Table 2. The difference in the final time, Δt_f, shown in the table, which is the difference between the on board Julian date and the true Julian date at the final time in seconds, shows that the error at the end of the run is under 100 milliseconds for 3 of the runs, but that it is slightly over for case 3 with a 2 second bias. This value changes between runs due to the performance of the filter, the different realizations of noise, and a different initial a priori offset from the truth. These values are only meant to serve as examples that show the goal for time bias estimation is within reach.

Mask R-CNN Detections Results

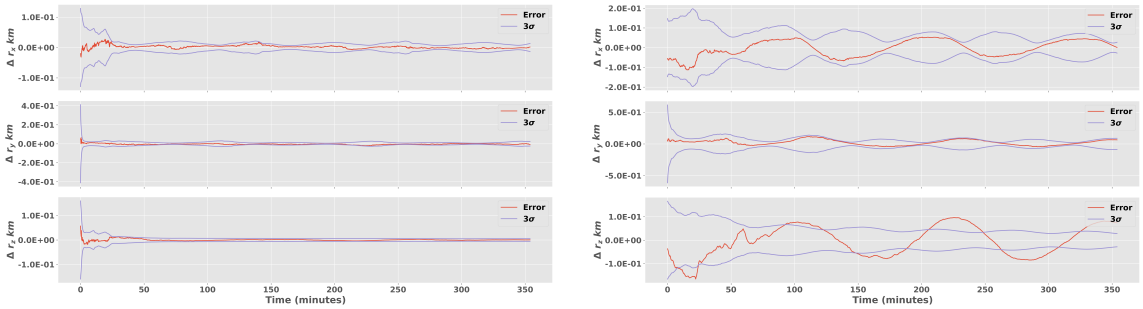
While the tests using simulated detections provide a method of inspecting the filter performance without the detector in the loop, the tests outlined in this subsection include cases using the simulated image generation and the crater detection for an end to end software evaluation. Figure 8 compares the results from the simulated detections and the Mask R-CNN detection outputs for a 3-orbit test case with initial conditions matching Test Case 0 in Table 1 with dynamics models matching the previous simulation setup for both the true trajectory and the CNT time update model. These plots reflect the differences between the crater detection inputs that result in larger state errors with the detector in the loop. In this section, the results will only include position errors and their

Table 3: Root mean square error for time bias tests and final time difference ($t_{f,true} - \hat{t}_f$)

Cases	RMS_x (km)	RMS_y (km)	RMS_z (km)	RMS_{3D} (km)	Δt_f (ms)
Case 0, 2 second bias	0.009	0.01	0.005	0.014	- 99.840
Case 0, 20 second bias	0.012	0.017	0.006	0.021	72.777
Case 3, 2 second bias	0.008	0.011	0.016	0.021	-137.426
Case 3, 20 second bias	0.01	0.011	0.016	0.022	25.864

respective 3σ covariances for clarity. Table 6 summarizes the resulting RMS values for these tests.

Figure 8 shows much larger errors with the detector in the loop compared to those of the simulated detections. The figure also shows that the mean position error exceeds the 3σ covariance bounds periodically. This result indicates that while the simulated detection outputs emulate the expected measurement error of the real detector, there are other potential differences between the construction of the simulated data to the detector outputs that are not accounted for in the current configuration. The primary advantage of using simulated detections is runtime, since it does not include the generating images and running the Mask R-CNN model, which are slow processes compared to the execution of the navigation filter. Improved modeling of the real detector through the simulated detection generation is planned for future work, however, differences in the number of detections and an observed residual bias are explored in the subsections below.



(a) Simulated detections test

(b) Detector in the loop test

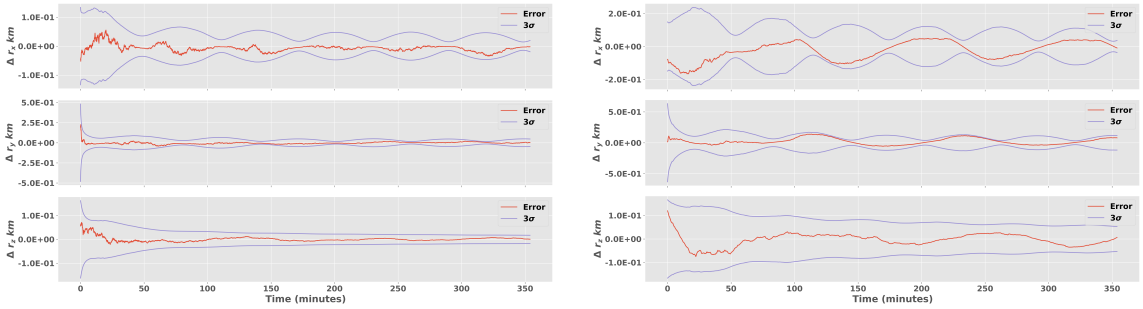
Figure 8: Position state errors (without setting a maximum number of craters processed or bias corrections)

Number of detections

The simulated detections contain more craters per image compared to the detector outputs. Since the simulated craters do not have an associated confidence threshold, all detections produced in the simulation would be processed by the estimation filter. In comparison, the detector's outputs are parsed down based on the threshold and if the identification method fails to find a matching crater, the set of detections will be further reduced. In some cases, the simulated detections include >100 craters in an image, whereas the detector might have around 10 or fewer craters.

To better compare the state errors between the simulated and real detections, a maximum number of craters per image can be set in the test scenario. This enables further inspection of the effect of fewer processed measurements in the estimation filter, where Fig. 9 shows the resulting state errors when only 3 measurements are used in each image. For the detector in the loop test, the detections

are sorted by confidence to keep the 3 highest confidence craters, but the detections must also pass through the 0.8 confidence threshold set in previous testing. Compared to the observed errors when all detections are processed, the resulting plots show a higher frequency periodic covariance change for both cases and the mean errors from processing the real detections are now bounded by the 3σ, indicating that the filter obtains a more consistent state solution.



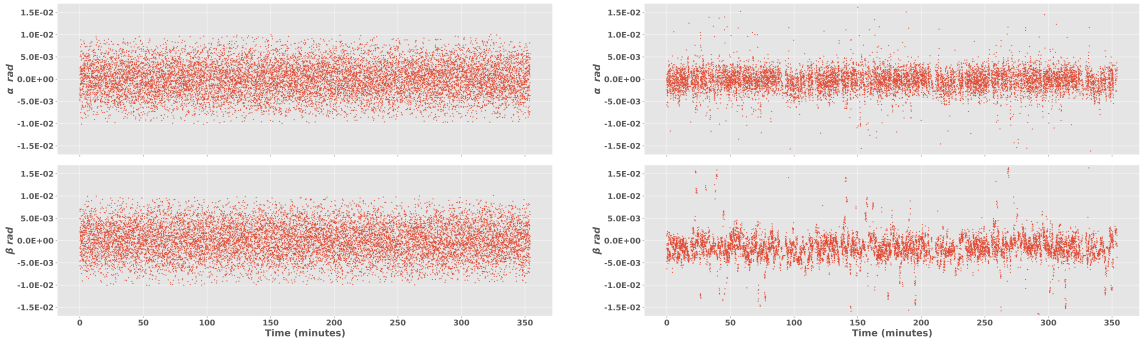
(a) Simulated detections test

(b) Detector in the loop test

Figure 9: Position state errors with a maximum of 3 detections per image, without bias correction

Measurement bias

Running a test case in which the true states are used as the filter's a priori mean enables inspection of potential measurement biases in the detector outputs. With the detections used to generate the previous test case in Fig. 9 and the true states, Fig. 10 shows the residual outputs for the simulated and real detector tests and the output statistics are shown in Table 4. The non-zero means for the prefit residuals in the detector test suggest that there are unmodeled measurement biases that may cause the larger state errors when compared to those from the simulated detections that have negligible residual means for α and β.



(a) Simulated detections test

(b) Detector in the loop test

Figure 10: Residual outputs with a maximum of 3 detections per image and using the true state as the filter a priori

While the source of the observed measurement bias requires further investigation, one method to reduce the error caused by the bias is to apply a correction to the observation model provided in Eq.

Table 4: Residual output statistics with a maximum of 3 detections per image and using the true state as the filter a priori

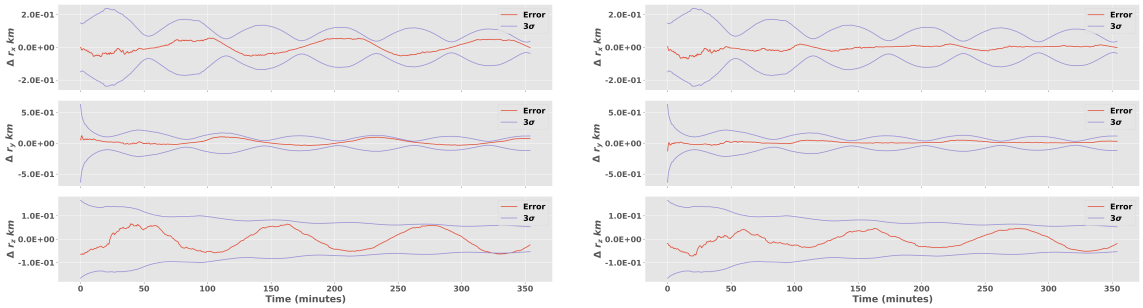
	$\bar{\alpha}$ (rad)	$\alpha\sigma$ (rad)	$\bar{\beta}$ (rad)	$\beta\sigma$ (rad)
Simulated detections	-1.01E-7	0.0038	-1.02E-5	0.0038
Mask R-CNN detections	-3.67E-4	0.0025	-0.0016	0.0027

1 and 2. This bias correction is applied to reprocess the Mask R-CNN detector outputs via

$$\alpha = \tan^{-1}\left(\frac{x_c - 0.5N}{f}\right) - \bar{\alpha} \text{ and} \tag{21}$$

$$\beta = \tan^{-1}\left(\frac{y_c - 0.5N}{f}\right) - \bar{\beta}, \tag{22}$$

where $(\bar{\alpha}, \bar{\beta})$ corresponds to the mean bias values from Table 4 for the respective bearing angles. After the bias correction is applied to the detector's observation outputs, the resulting errors show the improvements to the state error outputs in Fig. 11. Since the observed β bias is much larger than the α bias, the first test in Fig. 11a shows the resulting state errors when only the β correction is applied. After the bias correction is applied, the state errors are lower. In particular, the initial estimate of the Z-component is more accurate. However, additional periodic effects are also shown after the β bias correction. The second test in Fig. 11b shows further reduction in the errors when both α and β biases are subtracted, particularly by inspecting the amplitudes of the periodic variation compared to the errors when only the β correction is applied. Table 5 summarizes the residual statistics corresponding to the tests in Fig. 11, where the bias corrections effect can be observed primarily in the reduction of the β mean value.



(a) Detector in the loop test, with β correction only

(b) Detector in the loop test, with α and β correction

Figure 11: Position state errors with bias correction applied

In these tests, the measurement noise covariance matrix $\boldsymbol{R}_k$ was increased for the test cases using the Mask R-CNN detector. This was done to account for image generation issues encountered when using the LROC Global Maps for trajectory simulations. The issues with the image generation are likely contributing factors to the observed measurement biases that were previously discussed. While using the Robbins crater catalog together with the LROC maps enabled the automated dataset generation and training for the Mask R-CNN detector model, analysis with the current software

pipeline shows the need for further testing of this dataset to validate its use to generate simulated terrain images from an orbiting camera. This analysis is an ongoing effort, along with work towards integrating Blender and PANGU into the image generation pipeline to enable variations in lighting conditions, orbits, and spacecraft attitude that further test the robustness of the CNT methods outlined in this paper.

Table 5: Residual output statistics with a maximum of 3 detections per image, with bias correction

	$\bar{\alpha}$ (rad)	$\alpha\sigma$ (rad)	$\bar{\beta}$ (rad)	$\beta\sigma$ (rad)
Mask R-CNN detections with β correction	1.02E-4	0.0025	1.26E-6	0.0027
Mask R-CNN detections with α and β correction	9.01E-5	0.0025	3.85E-5	0.0027

Table 6: Root mean square error for detector in the loop tests

Cases	RMS_x (km)	RMS_y (km)	RMS_z (km)	RMS_{3D} (km)
Simulated detections, all craters	0.006	0.007	0.004	0.009
Simulated detections, 3 craters	0.014	0.013	0.010	0.022
Mask R-CNN, all craters	0.041	0.047	0.065	0.091
Mask R-CNN, 3 craters	0.061	0.056	0.027	0.087
Mask R-CNN, 3 craters, β correction	0.035	0.047	0.043	0.073
Mask R-CNN, 3 craters, α & β correction	0.017	0.022	0.031	0.042

CONCLUSION

The goal of this work is to demonstrate a proof of concept for semi-autonomous terrain relative navigation in lunar orbit with under 100 meters of error in an estimated position solution and under 100 milliseconds of error in an estimated time solution using data and algorithmic constraints compatible with implementation on a small satellite form factor. Current results indicate that under ideal conditions, a position solution within this error threshold is achievable. The time estimation algorithms developed for this work show in preliminary results that time bias estimation to this degree of accuracy is possible, but this capability does not yet meet requirements standard.

Future Work

While the CNT software is on track to meet its targets, there are several areas in which further work is needed. First, bias estimation algorithms must be improved. The simulation of the random walk and the time bias when coupled with the existing CNT simulation and filtering framework must be investigated further and compared to real-world bias estimation paradigms. The introduction of more realistic models of clock bias and of random walk in the timestamps would provide added realism to the project and allow for improved results when modeling time bias alongside errors in the measurements and dynamical perturbations. Additionally, further work is currently underway to build and run the CNT software on a NVIDIA Jetson computer, a small satellite compatible piece of hardware. Additional hardware and software constraints will be applied to the software build in order to increase runtime performance. Improving the software's ability to compensate for the detector bias is also a key component of planned work in the short-term future. In addition to this,

improving the data the detector is receiving is important to improving the simulation's realism and to better estimate potential real-world performance.

ACKNOWLEDGEMENTS

Work completed at The University of Texas at Austin was conducted under NASA cooperative agreement 80NSSC20M0087. The authors would like to thank Alexander Chiu for his contributions to the CNT software presented in this work.

REFERENCES

[1] F. Chandler, D. Israel, and S. A. Townes, "NASA Technology Roadmaps: TA 5: Communications, Navigation, and Orbital Debris Tracking and Characterization Systems," Tech. Rep. 20546-0001, National Aeronautics and Space Administration (NASA), NASA Headquarters, Washington DC, 2015.

[2] F. C. Hanak, *Lost in low lunar orbit crater pattern detection and identification*. PhD thesis, The University of Texas at Austin, 110 Inner Campus Drive, MAI 101, Austin, TX 78712, 5 2009.

[3] L. M. Downes, T. J. Steiner, and J. P. How, "Neural Network Approach to Crater Detection for Lunar Terrain Relative Navigation," *Journal of Aerospace Information Systems*, 2021, pp. 1–13.

[4] E. Emami, G. Bebis, A. Nefian, and T. Fong, "Automatic Crater Detection Using Convex Grouping and Convolutional Neural Networks," 12 2015, pp. 213–224, 10.1007/978-3-319-27863-6_20.

[5] S. G. Catalan, J. S. McCabe, and B. A. Jones, "Implementation of Machine Learning Methods for Crater-based Navigation," *AAS/AIAA Astrodynamics Specialist Conference*, No. AAS 21-519, Big Sky, MT Virtual, August 2021.

[6] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, Vol. 32, 2019, pp. 8026–8037.

[7] J. Bai, F. Lu, K. Zhang, *et al.*, "ONNX: Open Neural Network Exchange," `https://github.com/onnx/onnx`, 2019.

[8] E. Speyerer, M. Robinson, B. Denevi, *et al.*, "Lunar Reconnaissance Orbiter Camera global morphological map of the Moon," *Lunar and Planetary Science Conference*, No. 1608, 2011, p. 2387.

[9] S. J. Robbins, "A New Global Database of Lunar Impact Craters >1–2 km: 1. Crater Locations and Sizes, Comparisons With Published Databases, and Global Analysis," *Journal of Geophysical Research: Planets*, Vol. 124, No. 4, 2019, pp. 871–892, https://doi.org/10.1029/2018JE005592.

[10] S. Parkes, I. Martin, M. Dunstan, and D. Matthews, "Planet surface simulation with PANGU," *Space ops 2004 conference*, 2004, p. 389.

[11] Blender Online Community, *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[12] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, Vol. 5, No. 1, 1957, pp. 32–38.

[13] K. G. Murty, "Letter to the Editor—An Algorithm for Ranking all the Assignments in Order of Increasing Cost," *Operations Research*, Vol. 16, No. 3, 1968, pp. 682–687.

[14] B. A. Jones, "Surface Feature Navigation in Low Lunar Orbit," *18th Annual AAS/AIAA Space Flight Mechanics Meeting*, Galveston, Texas, January 28 - 31, 2008.

[15] R. J. Carpenter and C. N. D'Souza, "Navigation Filter Best Practices," Tech. Rep. TP–2018–219822, National Aeronautics and Space Administration (NASA), NASA Engineering and Safety Center, Hampton, Virginia, April 2018.

[16] B. E. S. Byron D. Tapley and G. H. Born, *Statistical Orbit Determination*. Elsevier Academic Press, 1st ed., 2004.

[17] W. Folkner, J. Williams, and D. Boggs, "The Planetary and Lunar Ephemeris DE 421," *Interplanetary Network Progress Report*, Vol. 42–178, August 2009.

[18] E. M. Standish, "The Planetary and Lunar Ephemerides DE/LE 405," *Interoffice Memorandum*, August 1998.