(Preprint) AAS AE6505 Final Project

SPACECRAFT POSITION ESTIMATION IN LEO USING TERRAIN RELATIVE KALMAN FILTERING TECHNIQUES

Harrison Jenkins*

In the following work, we leverage non-linear Kalman Filtering algorithms to estimate the position of a spacecraft in the ECI (Earth-Centered Inertial) frame with sub-kilometer accuracy. To our knowledge, this represents the first time terrain-relative Kalman Filtering techniques have been applied to a vehicle in Low Earth Orbit. This paper uses both Extended and Unscented Kalman Filters to achieve position estimation based solely on visual inspection of a known lake. These filters will employ the two-body equation of motion to model the spacecraft's nominal trajectory (accounting for both the J2 perturbation and atmospheric drag losses) and leverage observations of a known lake coastline to refine the spacecraft's position estimate. This paper will explore how the magnitude of uncertainty in the initial state affects the filters' ability to locate the satellite.

Ultimately, our work will show that both position estimation error and covariance diminish exponentially during the flyover period. Further, while both the UKF and EKF implementations show clear performance benefits and sub-kilometer position estimations, the UKF outperforms the EKF overall.

INTRODUCTION

Accurately understanding the location of a satellite is critical for the success of any mission. Precise knowledge of a spacecraft's position enables more reliable communication with the ground and better planning for downlinks and safety maneuvers. While most satellites, especially those in Low Earth orbit (LEO), utilize GNSS (Global Navigation Satellite System) to localize themselves in orbit, this service is not always available. Such GNSS denied regions may be created by a satellite's own communication interference, "jamming" efforts from a hostile spacecraft or military operation, or a simple lack of coverage (especially in remote oceanic or desert environments).

In such regions, most satellites fall back on odometry, communication with their ground station, and (if applicable) communication with other satellites in their constellation. While these present strong localization for brief GNSS dropouts, there may be cases where no communication with the ground or other satellites is possible for long periods of time. In cases like these, odometry will eventually accumulate large position errors, and visual localization techniques become an attractive complement. As such, we turn to **Terrain Relative Navigation**, a method of leveraging observations of surrounding terrain to localize the spacecraft. While Terrain Relative Navigation is well studied for low altitude cases like aircraft and planetary entry, few studies examine the use of Terrain Relative navigation for spacecraft in orbit. This is a particularly challenging task because it involves relatively high speed and high altitude, causing measurements to be more sparse and less

*PhD Student, Georgia Tech Daniel Guggenheim School of Aerospace Engineering, Space Systems Design Laboratory.

resolved. Notable studies include Maggio et al., who use EKF and smoothing techniques on a high altitude balloon and sub-orbital rocket[1] and Critchley-Marrows et al, who use a combination of star and horizon trackers to locate GNSS denied spacecraft in LEO.[2] To the best of our knowledge, the following study is the first to use a purely terrain-based filtering approach for position estimation in LEO. We aim to achieve sub-kilometer precision estimations at a bare minimum; however, we hypothesize that we can achieve accuracy on par with the aforementioned studies (100-300m).

Problem Definition

For this study, we take a standard 6U cubesat (dimensions $20 \times 10 \times 30$ cm) to be our vehicle of interest. This gives us a maximum cross-sectional area of 0.06 m^2, and a mass anywhere between 5-20 kg (for our study we choose 12 kg). We let the spacecraft's sensor have a FOV = 10°, frame rate of 30Hz, and output dimensions of (H=256,W=320) in pixels.

We begin the experiment at 2:00 GMT with the spacecraft located at $r_{true} = \begin{bmatrix} 3477.4 & -3719.3 & 4698.1 \end{bmatrix}^T$ km with an initial velocity of $v_{true} = \begin{bmatrix} 5.5407 & 5.1803 & 0 \end{bmatrix}^T$ km/s. These initial conditions position the vehicle for a direct fly-over of Seneca Lake, NY, our target of interest. This experiment assumes that, at t_0, the spacecraft has successfully identified Lake Seneca and pulled the corresponding catalog entry. With these initial conditions and the camera intrinsics above, we get a ground swath of 100km, leaving Lake Seneca in view for just 7 seconds. Given this, upon successfully identifying the Lake in its FOV, the spacecraft now knows its position to within a 100km radius. The objective of the experiment is to use the aforementioned filtering techniques to refine its position estimation to within 1 km. For each experiment, we randomly initialize the position of the satellite in a 100km bounding box and run our algorithms to test how well it converges over the flyover period (see the **Results** Section for tests with various noise magnitudes).

The rest of the paper will be outlined as follows. The next section, **Nomenclature**, we give an overview of the important variables referenced throughout this paper. Following that we explain our **Methodology**. This section is details the EKF and UKF algorithms we use for state estimation, as well as the non-linear system dynamics and measurement model we adopt. Following this we present our **Results**. After this, we detail the findings, limitations, and implications of our work in the **Discussion** section. Lastly, we provide a **Conclusion**, **Acknowledgment**, and **References**.

NOMENCLATURE

1. P_k^- = Estimation covariance after the time update.
2. X_k^- = State estimate after the time update.
3. P_k^+ = Estimation covariance after the time update.
4. X_k^+ = State estimate after the measurement update.
5. ω= Noise realization.
6. A= Jacobian of system dynamics.
7. F_k= State transition matrix at time t_k.
8. H= Jacobian of the measurement model.
9. Q= Process noise. Uncertainty in system dynamics.
10. R= Measurement noise. Uncertainty in measurements.
11. y= Measurement of the true state.
12. Δy= Measurement residuals.
13. K_k = Kalman Gain.

METHODOLOGY

Overall, this study employs numerical simulation for spacecraft dynamics and non-linear transformations of coastline coordinates to generate measurements for each timestep. The specific dynamic models and transformations will be explained in further detail in the ensuing paragraphs of this section. First, however, we give background to the filtering algorithms themselves and discuss them in the context of this experiment.

Extended Kalman Filter (EKF)

This study will implement a hybrid Extended Kalman Filter. In other words, while the system dynamics are continuous, the measurements come at discrete time intervals. For this simulation, the measurement frequency is determined by the frame rate of our camera (30 Hz).

In general, the EKF is a linearized version of the original, linear Kalman Filter put forth by Kalman in 1960.[3] This filter works by approximating the system's dynamics and measurement model as linear at each timestep of the simulation. Importantly, the EKF performs this linearization about the previous state estimation, rather than the nominal trajectory (computed purely from odometry).[4] The mathematical framework for the EKF is as follows :

1. $x_k^- = f\left(x_{k-1}^+\right) + \omega$.
   Where $f(x)$ is the non-linear dynamic model for the system. This function will be discussed in further detail in **Time update: Simulating Orbit Mechanics**. In this study, we do not directly inject noise into the time update, rather we use separate dynamic models for the true state and estimated state. This naturally creates a noise realization (see **Time update: Simulating Orbit Mechanics**e for more details). Given this, $\omega = 0$ for our purposes.
2. $P_k^- = F_k P_{k-1}^+ F_k^T + Q$ Where $F_k = e^{A(t_k - t_{k-1})}$
3. $\Delta y = y - h(x_k^-)$
   Where $h(x)$ is the non-linear measurement model for the system. Both $h(x)$ and y will be discussed in further detail in **Measurement Update: Coordinate Transformations**.
4. $K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$
5. $X_k^+ = X_k^- + K_k \Delta y$
6. $P_k^+ = (I - K_k H) P_k^-$
7. $X_{k+1}^- = X_k^+$
8. $P_{k+1}^- = P_k^+$

Unscented Kalman Filter (UKF)

The Unscented Kalman Filter is generally an improvement on the Extended Kalman Filter for highly non-linear systems as it uses the *Unscented transform* which preserves the system's statistics up to the 3rd order.[5] Critically, the unscented transform applies the full non-linear transformation to the state rather than approximating it as linear near the previous state estimation. In the context of the UKF, we use $2n$ "sigma points" which are computed from $x_\sigma = \bar{x} \pm \sqrt{nP}$. Where n is the number of states in the previous state estimate, $\bar{x}$. Once the sigma points are computed, the nonlinear transformation is applied to each one and their mean is computed.

The general UKF algorithm is outlined below:[6]

1. Compute sigma points with: $x_{k-1,\sigma}^- = \bar{x}_{k-1}^+ \pm \sqrt{nP}$

2. Put each point through the nonlinear model: $x^-_{k,\sigma} = f\left(x^+_{k-1,\sigma}\right) + \omega$.
3. Compute the mean of the sigma points: $\bar{x}^-_k = \frac{1}{2n}\sum_1^{2n} x^-_{k,\sigma}$
4. Compute covariance: $P^-_k = \frac{1}{2n}\sum_1^{2n}(x_{k,\sigma,i} - \bar{x}_k)(x_{k,\sigma,i} - \bar{x}_k)^T + Q$
5. Compute new sigma points based on the updated state estimation and covariance: $x^-_{k,\sigma} = \bar{x}^-_k \pm \sqrt{nP}$
6. Put each sigma point through the non-linear measurement model to obtain estimated measurements $z = h\left(x^-_{k,\sigma}\right)$.
7. Compute the mean of sigma point measurements $\bar{z} = \frac{1}{2n}\sum_{i=1}^{2n} z_i$
8. Obtain a measurement of the true state: y
9. Compute the measurement covariance $P_z = \frac{1}{2n}\sum_{i=1}^{2n}(z_i - \bar{z})(z_i - \bar{z})^T$
10. Compute the cross covariance $P_{x,z} = \frac{1}{2n}\sum_{i=1}^{2n}[(z_i - \bar{z})(x^-_{k,\sigma} - \bar{x}_k)^T] + R$
11. Compute Kalman Gain: $K_k = P_{x,z}P_z^{-1}$
12. Compute final state estimation: $x^+_k = x^-_k + K_k(y - z)$
13. Compute final estimation covariance: $P^+_k = P^-_k + K_k P_z K^T_k$

Time update: Simulating Orbit Mechanics

This study uses two variations of the 2-body equation of motion as the dynamic model for our filters. More specifically, we choose to model the true state with the J2 perturbation and atmospheric drag losses taken into account, while the model for the estimator uses neither. The equations for each case are shown here:

Estimation model: Pure 2-body motion

$$\ddot{r} = -\frac{\mu}{r^3}\vec{r} \tag{1}$$

True State model: 2-body motion with J2 and Drag

$$\ddot{r} = -\frac{\mu}{r^3}\vec{r} + \frac{-3J_2\mu R_E^2}{2r^5}\begin{bmatrix} x(1-\frac{5z^2}{r^2}) \\ y(1-\frac{5z^2}{r^2}) \\ z(3-\frac{5z^2}{r^2}) \end{bmatrix} + \frac{-C_d A\rho v}{2m}\vec{v} \tag{2}$$

In our problem, we choose to model the satellite as a standard 6U cubesat in circular orbit at 550km above the Earth's surface. As such, we set: The coefficient of drag as C_d=2.5, the cross-sectional area as $A = 6e^-8\ \text{km}^2$, the mass as $m = 12\text{kg}$, and the atmospheric drag as $\rho = 0.25\frac{\text{kg}}{\text{km}^3}$. We also set the equatorial radius of Earth as $R_E = 6378.1\text{km}$, the gravitational parameter as $\mu = 3.986e5\frac{\text{km}^3}{s^2}$, and the J_2 coefficient as $J_2 = 1.0826e^-3$.

Using these models give us the following state vector: $\vec{X} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$.

With the model established we can now derive the system Jacobian required for the EKF implementation. This Jacobian will be used each timestep of the algorithm to get the state transition matrix, F, as outlined in **Extended Kalman Filter (EKF)**. Since the estimator uses **Eq. (1)** to compute the time updates, we simply compute the gradient of this equation with respect to the state

vector. As such, we derive the Jacobian in the following manner:

$$\vec{X} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \rightarrow \dot{\vec{X}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{\mu}{r^5} \begin{bmatrix} 0 & 0 & 0 & \frac{r^5}{\mu} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{r^5}{\mu} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{r^5}{\mu} \\ 3x^2 - r^2 & 3xy & 3xz & 0 & 0 & 0 \\ 3yx & 3y^2 - r^2 & 3yz & 0 & 0 & 0 \\ 3xz & 3zy & 3z^2 - r^2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

Thus, the Jacobian becomes:

$$A_{jac} = \frac{\mu}{r^5} \begin{bmatrix} 0 & 0 & 0 & \frac{r^5}{\mu} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{r^5}{\mu} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{r^5}{\mu} \\ 3x^2 - r^2 & 3xy & 3xz & 0 & 0 & 0 \\ 3yx & 3y^2 - r^2 & 3yz & 0 & 0 & 0 \\ 3xz & 3zy & 3z^2 - r^2 & 0 & 0 & 0 \end{bmatrix}$$

The use of a more accurate orbit model for the true trajectory and a simplified model for the estimator, allows noise to naturally enter the system without being injected directly into the filters. We illustrate the resulting modeling errors (which we treat as process noise) in **Figures (1) and 2**). Furthermore, since the process noise matrix, Q, represents the uncertainty in the estimator's model, we can compute Q analytically, and thus eliminate a set of hyper-parameters from our tuning process. We compute Q by simply taking the mean standard deviation of the difference between $x_{\text{est},k}$ and $x_{\text{true},k}$, over a single timestep, we get the diagonal elements of Q. In other words:

$$q_{\text{diag}} = \text{STD}(\text{mean}(f(x_{\text{est},t_0 \rightarrow t_1}) - f(x_{\text{true},t_0 \rightarrow t_1})))_{nx1} \rightarrow Q = \text{diag}(q_{\text{diag}})_{nxn}$$

.

Measurement Update: Coordinate Transformations

Computing the measurement update is the critical part of any Kalman filter. For this problem, we will simulate measurements by creating a projection of the target's known coastline coordinates onto the camera's lens based on the satellite's current state vector. In other words, we predict what the lake would look like if the satellite was viewing it from a certain position and orientation. We used this process to generate both the "true" measurements, y, and the predicted measurements, z.

Since we do not estimate pose in our filter, we must first decide how to define the satellite's pose based on this position and velocity estimation. For this study, we choose to have our satellite point directly Nadir with the top of its FOV aligned with the velocity vector (i.e. the along-track direction). With these assumptions we can compute the rotation matrix needed to convert any ECI coordinates to a unique quaternion with:

$$R = \begin{bmatrix} \vec{x} & \vec{y} & \vec{z} \end{bmatrix}$$

where

$$\vec{z} = \frac{-\vec{r}}{||\vec{r}||}; \quad y = \frac{-\vec{r} \times \vec{v}}{||\vec{r} \times \vec{v}||}; \quad \vec{x} = \frac{\vec{y} \times \vec{z}}{||\vec{y} \times \vec{z}||}$$

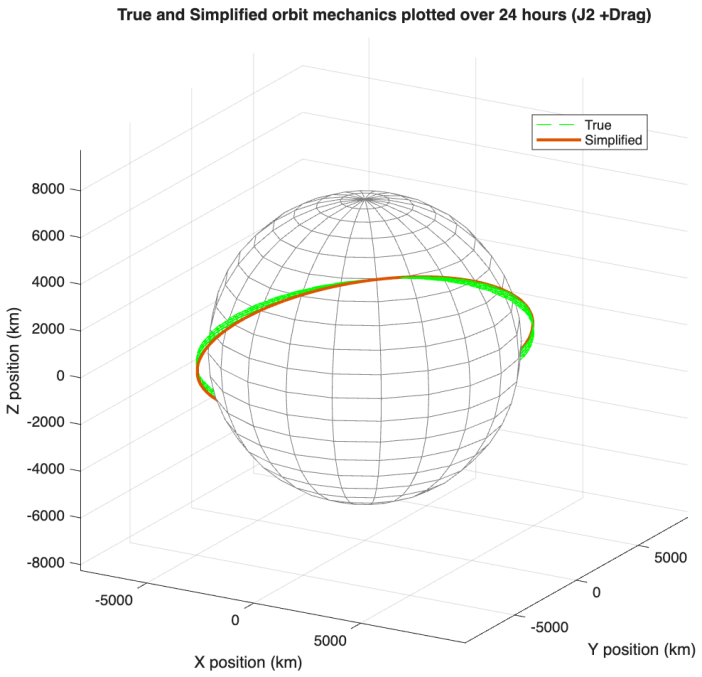


Figure 1: True and Estimator trajectories plotted over a 24 hour period

Then using Python's SciPy library, we convert the rotation matrix into a quaternion.

With a unique quaternion and state, we can proceed with the coordinate projection. To begin we leverage python's pyproj library to transform each point from Latitude, Longitude, and altitude coordinates to coordinates in the ECEF (Earth Centered Earth Fixed) frame. This library treats the globe as a WGS 84 Ellipsoid.[7] Once the points are in the ECEF frame, we define rotation matrices to convert from ECEF to ECI, then from ECI to the satellite's body frame, and lastly, from the body frame to the camera. This is computed as follows:

First we pick a reference time to fix the angle between the ECEF and ECI frames. For our study we choose 2:00 GMT for all computations. Implicit here, is the assumption that the Earth is stationary during time period of our simulation. In certain case this may not be a valid aussption, however, the estimation period for our study is roughly 7-10 seconds, thus this assumptions is acceptable for us. With this we can compute the rotation angle between the ECI and ECEF frames as:

$$\theta_{geo} = (2 \text{ GMT})\omega_{earth}$$

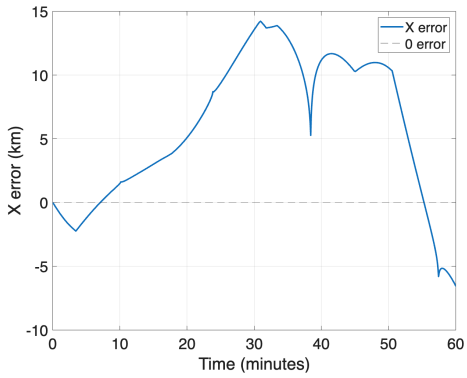
where $\omega_{earth} = 0.261799387\frac{rad}{s}$ is the Earth's rotation rate.

Then we can define the rotation matrix from ECEF to camera space using **Eq. (3)**:

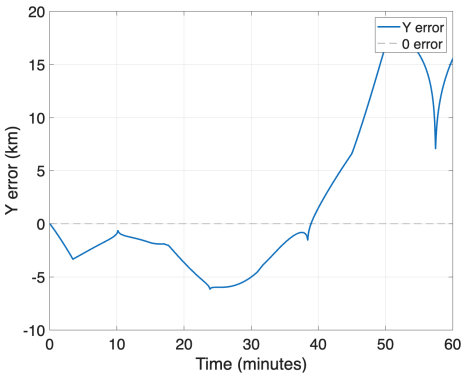
$$R_{\text{ecef}\to\text{cam}} = (R_{\text{body}\to\text{cam}})(R_{\text{eci}\to\text{body}})(R_{\text{ecef}\to\text{eci}}) \tag{3}$$

Where:

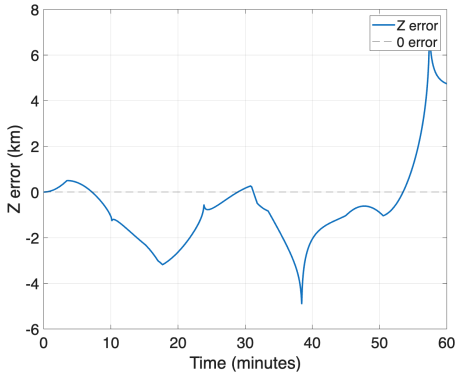
$$R_{\text{ecef}\to\text{eci}} = \begin{bmatrix} \cos(\theta_{geo}) & -\sin(\theta_{geo}) & 0 \\ \sin(\theta_{geo}) & \cos(\theta_{geo}) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



(a) X position error



(b) Y position error



(c) Z position error

Figure 2: Estimator modeling errors over a 1 hour period (ECI Frame)

$$R_{\text{body}\to\text{cam}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and $R_{\text{eci}\to\text{body}}$ comes from the quaternion.

With $R_{\text{ecef}\to\text{cam}}$, we can transform the ECEF coordinates to the camera frame using **Eq.(4)**:

$$\vec{X} = \vec{X}_{\text{ecef,Coordinate}} - \vec{X}_{\text{ecef,Sat}} \quad \to \vec{X}_{cam} = (R_{\text{ecef}\to\text{cam}})\vec{X} \tag{4}$$

Then we transform the camera coordinates, $\vec{X}_{cam}$, to 2D pixel-space coordinates using **Eq.(5)**:

$$u = (K_{1,1})x + (K_{1,3}) \quad v = (K_{2,2})y + (K_{2,3}) \tag{5}$$

where

$$x = \frac{x_{cam}}{z_{cam}} \quad y = \frac{y_{cam}}{z_{cam}} \quad K = \begin{bmatrix} f_x & 0 & \frac{\text{width}_{\text{cam}}}{2} \\ 0 & f_y & \frac{\text{height}_{\text{cam}}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

,

$$f_x = f_y = \frac{\text{width}_{\text{cam}}}{\tan(\frac{\text{fov}}{2})}, \quad \text{width}_{\text{cam}} = 320, \quad \text{height}_{\text{cam}} = 256, \quad \text{fov} = 10^\circ$$

.

Using this formulation, we can convert each coastline coordinate in our catalog of Lake Seneca into a synthetic remote view (see **Figure 3b**).

(a) Landsat image (b) Synthetic projection

Figure 3

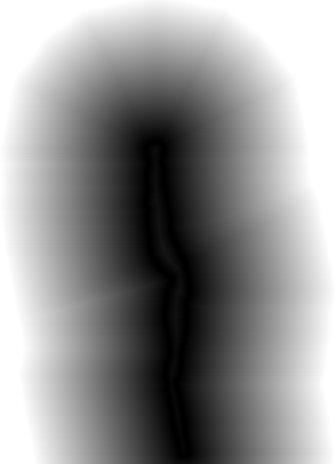


Figure 4: Example distance transform of Lake Seneca

In our algorithm, each synthetic view is represented as a vector of pixel coordinates (called a contour) which can be processed in OpenCV. For proper filtering we choose our measurements to be: **Centroid location** (for translational shifts), **Area** (for altitude shifts), **Distance transform** (which encompasses rotation, translation, and altitude differences).

The OpenCV's distance transform function computes the distance from each point in pixel-space to the nearest contour point. This yields a new image where each point's intensity corresponds to its distance from the nearest contour point (see **Figure 4**). This transformation is ideal when performing terrain relative navigation based on contours because it eliminates the need for point-wise correspondences between the true state measurement and the estimated state measurement. For this measurement, we first compute the distance transform for the true state and then we sample that transform using the coordinates of each point on the estimated contour. This yields a distance vector of the same length as the estimated contour. We then take the mean of this distance vector as one measurement. This alone however is insufficient for a state estimation filter however, as this distance is un-signed. In order to fix this, we compute the opposite distance vector (ie. sample the distance transform of the estimated contour using points from the true contour) and add the mean of this vector to the initial distance vector.

In this way our full measurement vectors become:

$$\vec{y} = \begin{bmatrix} \text{mean}(\vec{d}_{true}) + \text{mean}(\vec{d}_{est}) \\ x_{\text{centroid,true}} \\ y_{\text{centroid,true}} \\ \text{area}_{\text{true}} \end{bmatrix}$$

$$\vec{z} = \begin{bmatrix} \text{mean}(\vec{d}_{true}) + \text{mean}(\vec{d}_{est}) \\ x_{\text{centroid,est}} \\ y_{\text{centroid,est}} \\ \text{area}_{\text{est}} \end{bmatrix}$$

With measurement vectors, we can now define the measurement uncertainty matrix, R. This matrix will be a diagonal matrix of dimension 4x4 where each diagonal entry corresponds to the expected fluctuations/noise in the corresponding measurement. In this study we do not analytically derive these uncertainty values, rather treat this matrix as a tunable parameter. We find:

$$R = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 50^2 \end{bmatrix}$$

to be an optimal value for this matrix. These measurements are in pixel-space, thus R has units of pixels and pixels^2.

Lastly, given the lack of closed form equations for our measurement model, we opt to numerically compute the measurement Jacobian, H, using the finite differences method. For the i-th state in the system we compute the following each timestep:

$$\vec{x}_{\text{new,i=2}} = \vec{x} + \begin{bmatrix} 0 & eps & 0 & 0 \end{bmatrix}^T$$

$$H_{i=2} = \frac{1}{eps}[\Delta y(\vec{x}) - [y - h(\vec{x}_{\text{new, i=2}})]]$$

Here, eps becomes a potential tuning parameter, but we find 0.1 to be a good value for our system.

RESULTS

Our study seeks to accurately estimate the position of a 6U Cubesat in a GNSS denied region. For our purposes we assume the satellite has a catalog of known lake coastlines and that it can readily identify a match between an observed lake and the proper catalog entry.[8] After identifying a match the satellite will know its position to within 100 km. From this point, we seek to refine its position knowledge to within a kilometer in the duration of the flyover.

To begin our study we first allow the spacecraft to randomly initialize its position estimate to within 15 km and its velocity to 1 m/s. These conditions present generous apriori state knowledge (given the actual position uncertainty of 100 km) but it provides a good starting case to display the efficacy of the filter. Using this random initialization and an initial covariance matrix, P_0, based on this uncertainty, we show that the spacecraft is able to localize itself to within 0.5 km using the UKF and 0.9 km using the EKF. Initial and final error magnitudes are recorded in **Table (1)** and the 3D trajectory for true state is plotted against the EKF, UKF, and nominal state estimations in **Figure (5)**

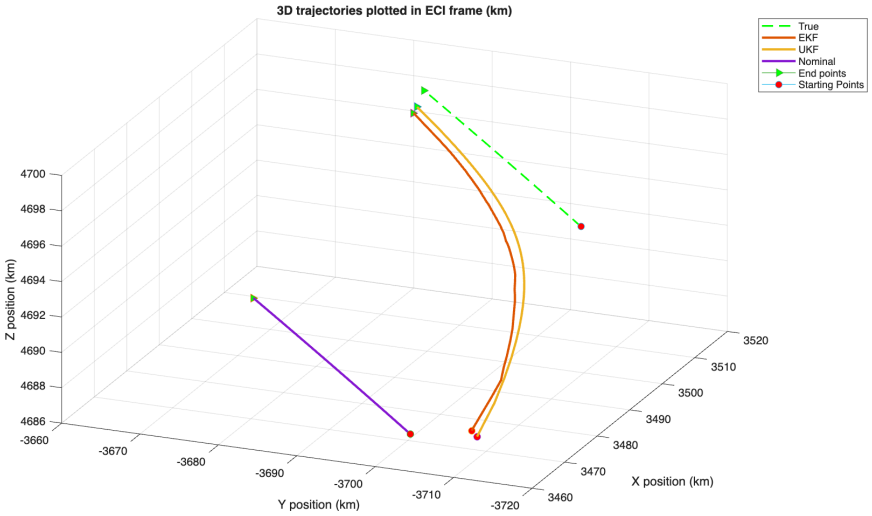


Figure 5: 3D Trajectory plots with initial uncertain of (15 km; 1m/s)

Table 1: Separation magnitudes with initial position and velocity uncertainy of 15km and 1m/s

State Estimator	Starting Separation (km)	Final Separation (km)
True	0	0
EKF	17.1804	0.87729
UKF	17.1075	0.4241
Nominal	23.1994	23.1994

Figure (5) shows exponential convergence toward the true state for both filters while the nominal trajectory remains at a near constant offset. These trends are made more apparent in **Figure (6)** which provides a breakdown of the error along each axis. From these plots, one will note that while both filters show sharp decline in estimation error, the UKF generally produces lower error than the EKF at each point in time. Also, one may notice that the nominal trajectory's error is not constant in time, by increases very slightly. This is a result of the noise realization discussed in **Time update**.

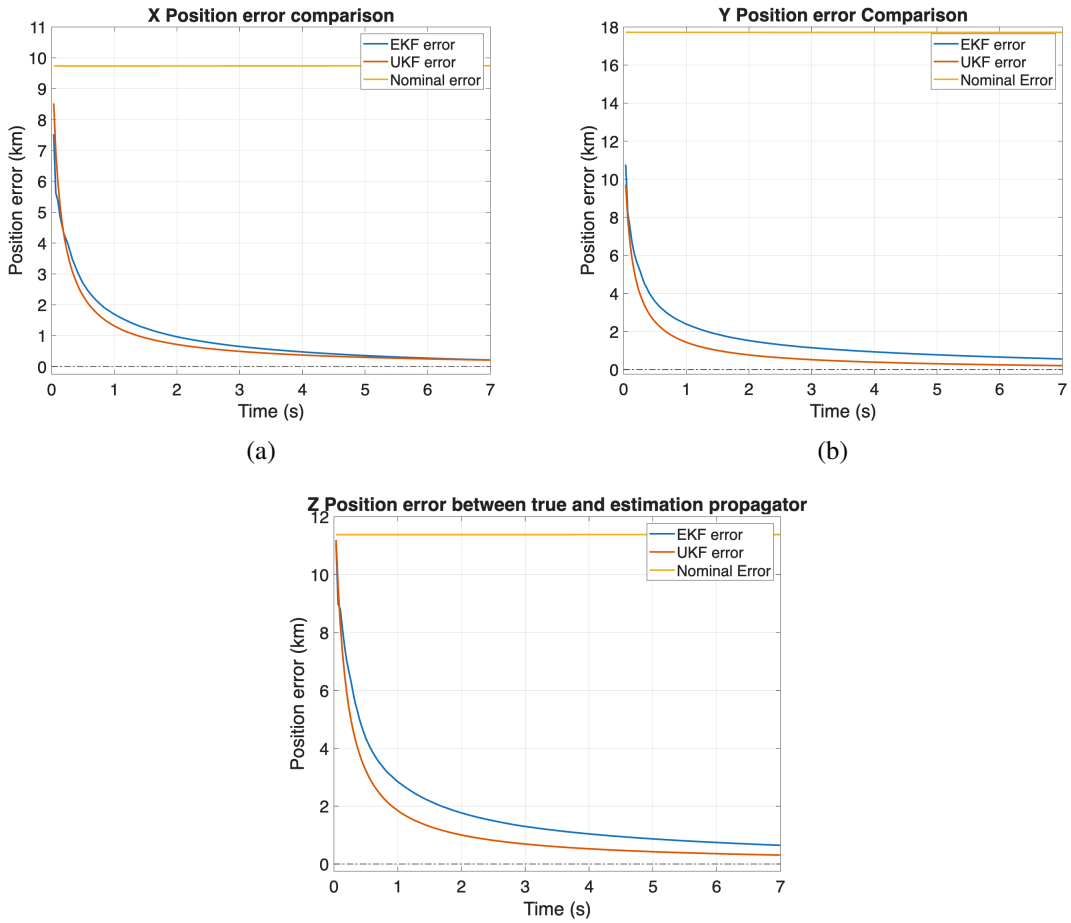
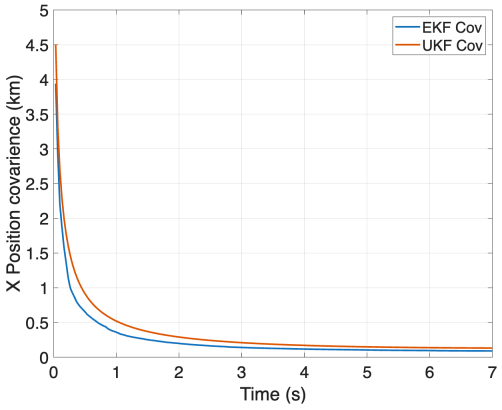


Figure 6: X, Y, and Z Error plots with initial uncertainty (15km; 1 m/s)

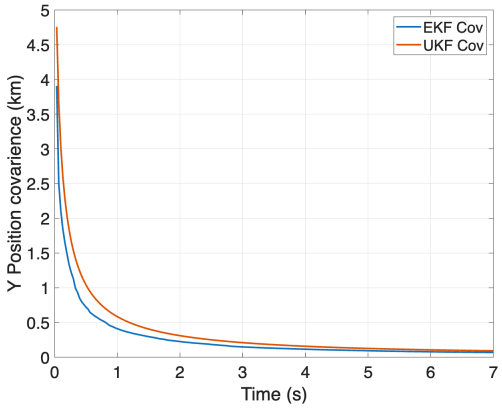
Examining the covariance plots yields yet more insight into the performance of the filters. Estimation covariance for the x, y, and z position of the satellite are recording in **Figure (7)**. From this figure we note that, while the UKF outperformed the EKF, the EKF claimed to have higher confidence in its estimations. This phenomena occurs because the EKF makes the assumption that the system is linear (locally) and builds its covariance matrix according to that assumption. The UKF however, accounts for the non-linearity to the third order and thus, it covariance is larger.[6]

While compelling, these results (as stated above) represent a generous apriori position and velocity knowledge. This is not generally realistic, thus we will incrementally relax the amount of apriori knowledge we start with to better understand these filters' ability to handle more realistic conditions. We begin by increasing the uncertainty in initial velocity from 1 m/s to 100 m/s. This yields the data in **Figure (8)** and **Table (2)**.

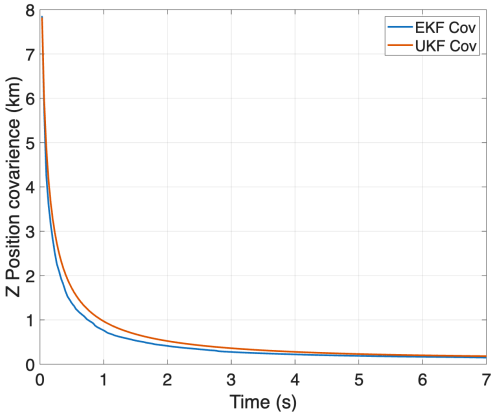
Finally we increase the uncertainty in initial position from 15 km to 70 km and leave the velocity uncertainty at 100 m/s. This yields the data in **Figure (9)** and **Table (3)**.



(a) X Covariance



(b) Y Covariance



(c) Z covariance

Figure 7: X, Y, and Z Covariance plots with initial uncertainty (15km; 1 m/s)

Table 2: Separation magnitudes with initial position and velocity uncertainy of 15km and 100m/s

State Estimator	Starting Separation (km)	Final Separation (km)
True	0	0
EKF	17.1798	1.5048
UKF	17.1397	0.68257
Nominal	23.293	23.293

Table 3: Separation magnitudes with initial position and velocity uncertainy of 70km and 100m/s

State Estimator	Starting Separation (km)	Final Separation (km)
True	0	0
EKF	33.9036	0.72224
UKF	42.1229	0.62102
Nominal	108.3355	108.3355

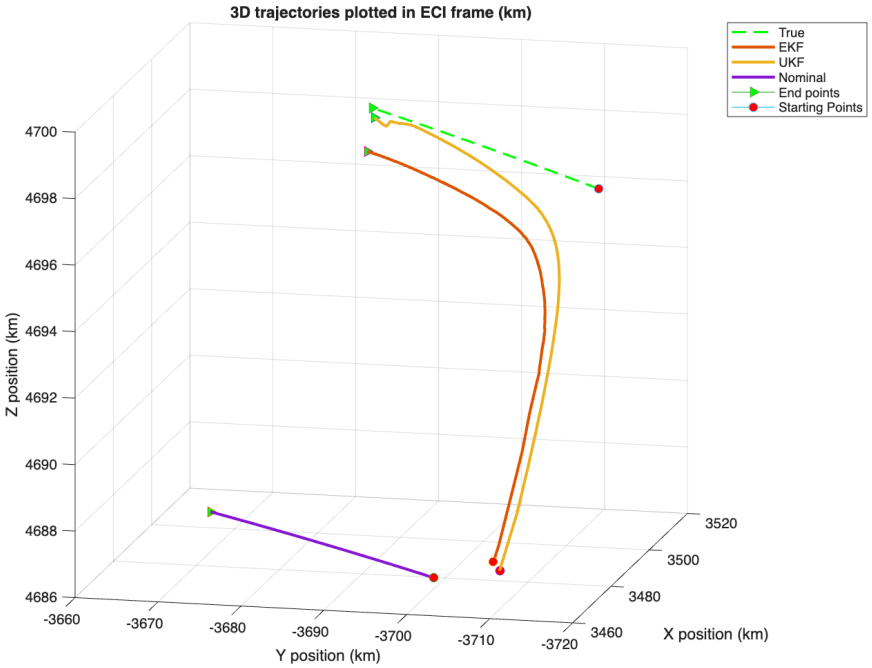


Figure 8: 3D Trajectory plots with initial uncertain of (15 km; 100m/s)

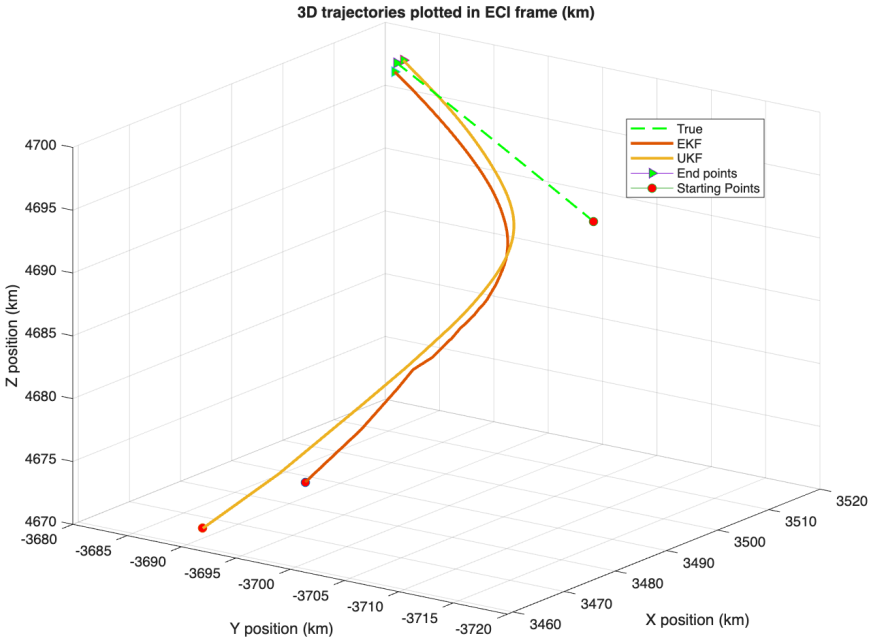


Figure 9: 3D Trajectory plots with initial uncertain of (70 km; 100m/s)

DISCUSSION

In this study we use non-linear Kalman Filters to reduce the state estimation error of our spacecraft by over 90% in all trials. We find that both the EKF and UKF provide similar decreases in order of magnitude, however the UKF outperforms the EKF in two main ways. First, the UKF yields

lower error state estimates in all trials. Secondly, the UKF produces more conservative covariance values. Furthermore, we find that the filter quickly asymptotes to a low error state regardless of the initial error magnitude.

Before continuing it is important to acknowledge the limitations of this work. First, the performance of our filters has been evaluated entirely in simulation, using projections of catalog coordinates to create synthetic views of our target. This creates a well controlled and consistent environment for implementing and comparing filters, however, it does not take into account the inevitable discrepancies between the catalog and the ever-changing coastline of the real lake.

Furthermore, the computation time associated with our current algorithm is not suitable for real-time position estimation. There is much room for optimization still however, the current version takes minutes to run a 7 seconds flyby at 30 Hz. That being said, we have found that decreasing the sampling rate to 10Hz or so does not signifigantly impact the UKF's position error. Thus with code optimization and a lower sampling rate, this algorithm should be able to perform real time estimation in the future.

Overall, we show that a satellite in low Earth orbit can achieve position accuracy between 400-700 m using the UKF and between 700m-1.5km using the EKF. Furthermore, we found that overall performance was not highly correlated with the error in the initial state as we expected in the beginning. In fact, the most accurate estimation for the EKF occurred with the highest initial error state. Similarly, the UKF's second most accurate estimation came in the trial with the greatest initial error. This result needs further testing to validate but it suggests that the filter performs can be used to disambiguate a satellite's position regardless of how poorly it is initially known after identifying a known target. This claim is substantiated by the fact that the upper bound of positional ambiguity is determined by the satellite's ground swath. In our case, simply identifying Lake Seneca in frame localizes the satellite to within 100 km of $42.8145°$N, $-76.9564°$W. Thus, as long as a known target is in view, the filter can localize the target to within a kilometer based on our results.

The covariances generated during the filtering process suggest a high confidence in the estimations. **Figure (7c)** shows position covariances on the order of 50-100m. This must be noted when using these filters, as they portray uncertainty that is less than the true position error produced in our trials. This is especially true of the EKF which consistently shows high errors and lower covariances than the UKF.

Lastly, while both filters remained relatively stable throughout the trials, the EKF showed signs of instability when initialized with high positional error. This should be noted when using these filters in the future.

Now we seek to contextualize this work with the current state of the art. It should first be noted that few studies use terrain relative navigation techniques to localize satellites in LEO. Most terrain relative position estimation takes place at far lower altitudes where ground objects can be observed with much greater resolution and the vehicle moves at slower speeds. Furthermore, most satellites use GNSS to localize themselves in LEO as it provides meter-level precision. Our study is intend to provide position estimation for spacecraft in GNSS denied regions however, thus terrain relative techniques remain attractive. While most the work in this field exists for low altitude vehicles (whether on Earth, Mars, or the moon), there are a few studies which use terrain identification to localize high altitude vehicles. Enright et al. report position accuracy of $<$300m for a Martian decent case study using a EKF based filter.[9] Maggio et al. present similar position errors when using EKF and smoothing techniques with a high altitude balloon and sub-orbital rocket.[1] While

the error presented in these studies is lower than what we achieve here, it is important to note that they operate at altitudes which are still lower than the 550 km used in this study. For this reason, we claim that our state estimators (the UKF in particular) holds up well with comparable state of the art.

There is much work still to be done and we identify the following as potential avenues for future work. 1) Tuning hyperparameters and further exploration to achieve a lower estimation error. 2) Extend the object identification to islands and other geographical features to provide more frequent estimation periods. 3) Code optimization to enable real-time estimation. 4) Connect this work with that of Carragher et al.[8] to continuously match ground observations to catalog entries and perform continuous state estimation. 5) Experiment with different pose assumptions for more flexibility.

CONCLUSION

In this study we have presented a novel position approach to GNSS denied localization using a purely terrain-relative filtering approach. We compare the performance of both a UKF and EKF and show that the UKF not only produces less localization error, but also converges more smoothly and outputs more accurate covariances. We show that our approach surpasses the initial goal of sub-kilometer accuracy, producing estimations with error between 400-700m. This frame work has the potential to provide real-time communication denied localization for spacecraft in LEO for the first time.

ACKNOWLEDGMENT

We thank Dr. Brian Gunter for mentorship and technical guidance in this research effort.

REFERENCES

[1] H. Fisher, G. F. Mendeck, *et al.*, "Vision-Based Terrain Relative Navigation on High-Altitude Balloon and Sub-Orbital Rocket," *arXiv preprint arXiv:2302.08011*, 2023.

[2] J. Critchley-Marrows and D. Mortari, "Vision-based navigation in low Earth orbit – Using the stars and horizon as an alternative PNT," *Advances in Space Research*, Vol. 71, No. 11, 2023, pp. 4802–4813, https://doi.org/10.1016/j.asr.2023.01.047.

[3] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, Vol. 82, 03 1960, pp. 35–45, 10.1115/1.3662552.

[4] *Nonlinear Kalman filtering*, ch. 13, pp. 393–431. John Wiley Sons, Ltd, 2006, https://doi.org/10.1002/0470045345.ch13.

[5] S. Julier, J. Uhlmann, and H. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, Vol. 45, No. 3, 2000, pp. 477–482, 10.1109/9.847726.

[6] *The unscented Kalman filter*, ch. 14, pp. 433–459. John Wiley Sons, Ltd, 2006, https://doi.org/10.1002/0470045345.ch14.

[7] National Geospatial-Intelligence Agency, "Department of Defense World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems," Tech. Rep. NGA.STND.0036_1.0.0_WGS84, National Geospatial-Intelligence Agency, Springfield, VA, USA, 2014. Latest revision available online.

[8] J. Carragher, A. Tyler, B. Gunter, D. Rowen, P. Leeds, and S. Cortes, "On-orbit detection, identification, and tracking of geographic targets with a long wave infrared camera," *Space Imaging Workshop*, Atlanta, GA, USA, October 7–9 2024.

[9] J. Enright, I. Jovanovic, L. Kazemi, H. Zhang, and T. Dzamba, "Autonomous Optical Navigation Using Nanosatellite-Class Instruments: A Mars Approach Case Study," *Celestial Mechanics and Dynamical Astronomy*, Vol. 130, No. 1, 2018, p. 13, 10.1007/s10569-017-9800-x.