MSFC-548 - Paper

Embedded Relative Navigation Sensor Fusion Algorithms for Autonomous Rendezvous and Docking Missions

Brandon K. DeKock, Avionics Engineer, bd Systems, Advanced Technology Division
Kevin M. Betts, Engineering Director, bd Systems, Advanced Technology Division
James H. McDuffie, Chief Engineer, bd Systems, Advanced Technology Division
Christine B. Dreas, GN&C Intern, bd Systems, Advanced Technology Division; School of Aerospace Engineering, Georgia Institute of Technology

BIOGRAPHY

Brandon DeKock received his Bachelor's of Mechanical Engineering from the University of Oklahoma in 2003, and his Master's of Aerospace Engineering from Georgia Tech in 2006. Kevin Betts received his Bachelor's of Mechanical Engineering from Georgia Tech and his Master's in Mechanical Engineering from Johns Hopkins University. James McDuffie has a BS in Physics from the University of Alabama-Huntsville (UAH) and MS in Physics and Electrical Engineering from UAH. Christine Dreas is currently an undergraduate student in Aerospace Engineering at Georgia Tech.

ABSTRACT

bd Systems (a subsidiary of SAIC) has developed a suite of embedded relative navigation sensor fusion algorithms to enable NASA autonomous rendezvous and docking (AR&D) missions. Translational and rotational Extended Kalman Filters (EKFs) were developed for integrating measurements based on the vehicles' orbital mechanics and high-fidelity sensor error models and provide a solution with increased accuracy and robustness relative to any single relative navigation sensor. The filters were tested through stand-alone covariance analysis, closed-loop testing with a high-fidelity multi-body orbital simulation, and hardware-in-the-loop (HWIL) testing in the Marshall Space Flight Center (MSFC) Flight Robotics Laboratory (FRL).

INTRODUCTION

The NASA Vision for Space Exploration requires an increased number of rendezvous and docking missions compared to previous NASA architectures with a goal of autonomous operations. Additionally, AR&D missions will be required in low lunar orbit, where GPS is unavailable as a high-fidelity absolute and relative position reference for both vehicles. The goal of this research was to develop a unified sensor fusion architecture to blend all available sensor measurements into a *single optimal state estimate* for use in the vehicle guidance and control system.

The sensor fusion algorithms accept measurements from a variety of relative navigation sensors with varying ranges and measurement types. Radio frequency (RF), light distance and ranging (LIDAR), and optical-based system measurements can all be incorporated in the sensor fusion algorithms. The translational filter uses the Clohessey-Wiltshire orbital relative motion equations as the core dynamics in the state transition matrix. The filters are augmented to also estimate sensor bias and scale factor errors for each individual active sensor in order to fully take advantage of multiple overlapping sensors. The filters were implemented with an option to run a Bierman U-D filter to ensure stable numerical performance for extended missions.

Fault Detection, Identification, and Recovery (FDIR) is a necessary feature of these filters because the sensors are capable of reporting anomalous measurements that could corrupt the filters' state estimates if not otherwise handled. Fault detection was implemented using a Chi-Squared test, whereby measurements that result in innovations that are too large are not incorporated into the state estimate. The filters also have the ability to recover if their estimate drifts so far from the measured values that the measured values are all rejected.

The filters were first implemented and tested using covariance analysis to determine the potential for increased accuracy. The filters were then implemented in the Simulation Package for Autonomous Rendezvous Test and Analysis (SPARTAN), a high-fidelity multi-body orbital simulation developed in MATLAB Simulink to study AR&D sensor and algorithm technologies. This integrated simulation allowed the sensor outputs to feed realistic guidance and control algorithms to fully analyze the affects of navigation accuracy on system performance. The robustness of the system was analyzed using Monte Carlo analysis for a variety of dispersed sensor and vehicle parameters.

The sensor fusion algorithms were also tested using processor and HWIL testing. To support this testing, the filters were autocoded from Simulink into C-code using MATLAB's Real Time Workshop. The filters were embedded on a PowerPC 750 Single Board Computer (SBC) running VxWorks to verify that computational performance was suitable for real-time implementation. HWIL testing using the filters was performed in the NASA MSFC FRL in August of 2007. This involved two short-range optical prototype sensors making relative position and relative attitude measurements of a set of docking targets mounted to a robotic arm that had 3-degree of freedom (DOF) translational and 3-DOF rotational capability. For the actual testing in the FRL, the filters were run as a dynamically linked library called by bdStudio, a visualization package developed by bd Systems that runs in a Windows environment. bdStudio was used to generate a pseudo-onboard view of the dockings, as well as a ground station-like third-person view of the dockings and display the raw sensor output as well as the filtered estimates.

Preliminary analysis of the filters indicates strong and robust performance. The open-loop covariance analysis confirmed that the errors and uncertainties were behaving as anticipated and that the filters were able to converge on the sensor error parameters. The errors were small enough that the simulated docking would be considered successful. Next, a Monte-Carlo analysis verified the closed-loop system performance of the filters with the guidance and control algorithms used in SPARTAN for a variety of dispersed sensor and vehicle parameters. The filters' errors were always small enough to allow for a successful docking, and no harmful interactions with the guidance and control algorithms were detected. The filters also remained numerically stable throughout all of the Monte Carlo runs. The filter timing performance in an embedded environment was characterized by running the filters in VxWorks on the PowerPC 750. It was determined that both filters can complete one execution in a mean time of 0.0041 seconds on an IBM750GX. This level of resource consumption verifies that these filters are credible for real-time applications in currently available flight hardware. During FRL HWIL testing, the filters were able to track the docking targets with acceptable errors through approach trajectories starting 30 meters from the sensors and ending 1 meter from the sensors. While this testing was done in an open-loop fashion, the error in the estimate was sufficiently small to allow for successful docking.

The test-verified embedded relative navigation filters are excellent candidates for onboard spacecraft relative navigation, as evidenced by their increased accuracy relative to any single sensor, their tolerance to bad measurements, and their ability to be run effectively on real-time hardware platforms. The next step in increasing the technology readiness level of the algorithm package will be through closed-loop HWIL testing followed by flight test opportunities.

KALMAN FILTER IMPLEMENTATION

The main goal of the filter implementation was to create the ability to optimally fuse data from all of the available sensors into a single state estimate that did not experience inappropriately large jumps when various sensors were turned on or turned off. In order to meet this goal, sensor error state estimates, in the form of measurement bias estimates and measurement scale factors were included for each sensor. This allows for filter innovations due to measurements from various sensors to be whitened as the bias estimates approach the correct value (assuming the sensor noise is Gaussian). A notable side benefit of this is that during a mission phase where two or more sensors are active, the more accurate sensor will drive the biases and scale factors of the less accurate sensor closer to the correct values, thereby performing live calibration of the less accurate sensor. This newly calibrated sensor is now better equipped to guide the spacecraft in case of a sensor outage among any of the other sensors.

Filter States

The states of the translational Extended Kalman filter can be divided into two groups; those describing the vehicle motion and those describing the sensor errors. The states describing relative vehicle motion are the relative positions and velocities of the chaser vehicle with respect to the target vehicle, expressed in the target vehicle's LVLH frame. The states describing the sensor errors are one bias state and one scale factor state for each scalar element of a measurement. The states of the rotational extended Kalman filter can also be divided in to those describing vehicle motion and those describing the error states of the sensors, with the angular scale factor term being a function of range to target rather than whole angular value.

General Equations

In the HYDRA Kalman filter implementation, both discrete linear and discrete extended Kalman filter equations are used. The discrete extended Kalman filter equations are provided in Table 1 below.

State Representation	$\underline{x}_k = \Phi_{k-1}\underline{x}_{k-1} + \Gamma_{k-1}\underline{u}_{k-1} + \gamma_{k-1}\underline{w}_{k-1}, \underline{w}_{k-1} \sim N(\underline{0}, Q_k)$
Measurement Model	$\underline{z}_k = \underline{h}_k(\underline{x}_k) + \underline{v}_k, \underline{v}_k \sim N(\underline{0}, R_k)$
Noise Correlation	$E[\underline{v}_k \underline{w}_k^T] = 0$ for all k, $E[\underline{v}_j \underline{v}_k^T] = [\underline{w}_j \underline{w}_k^T] = 0$ if $j \neq k$

State Propagation	$\hat{\underline{x}}_k^- = \Phi_{k-1}\hat{\underline{x}}_{k-1}^+ + \Gamma_{k-1}\underline{u}_{k-1}$
Cov. Propagation	$P_k^- = \Phi_{k-1}P_{k-1}^+\Phi_{k-1}{}^T + \gamma_{k-1}Q_{k-1}\gamma_{k-1}$
Kalman Gain Update	$K_k = P_k^- H_k(\hat{\underline{x}}_k^-)^T[H_k(\hat{\underline{x}}_k^-)P_k^- H_k(\hat{\underline{x}}_k^-)^T + R_k]$
State Update	$\hat{\underline{x}}_k^+ = \hat{\underline{x}}_k^- + K_k\underline{\varepsilon}_k$, $\underline{\varepsilon}_k = \underline{z}_k - \underline{h}_k(x_k^-)$
Covariance Update	$P_k^+ = [I - K_k H_k(\hat{\underline{x}}_k^-)]P_k^-$

Table 1: Discrete extended equations

Φ_k is obtained from the time-varying state dynamics vector F as follows (Reference 1):

$$\Phi_k = e^{\mathbf{F}\Delta t} = \mathscr{L}^{-1}[(\text{sI-F})^{-1}]$$

Γ_{k-1} is determined in a similar fashion:

$$\Gamma_k = e^{\mathbf{B}\Delta t} = \mathscr{L}^{-1}[(\text{sI-}\mathbf{B})^{-1}]$$

For our purposes, γ_{k-1} is an identity matrix and the state process noise $\underline{w}_{k-1}$ is the same size as the state vector in all implementations.

Numerical round-off problems can sometimes be encountered in the calculation of the covariance update. One solution to this is to use the Joseph algorithm, which assures positive-definite symmetric covariance matrices with a greater tendency towards numerical stability. It utilizes an alternate form of the covariance update equations as follows:

$$P_k^+ = [I - K_k H_k(\hat{\underline{x}}_k^-)]P_k^-[I - K_k H_k(\hat{\underline{x}}_k^-)]^T + K_k R_k K_k^T$$

for the discrete extended Kalman filter. The translational extended multi-sensor Kalman filter uses the version of the Joseph algorithm intended for extended Kalman filters. This was not done in reaction to numerical stability problems but simply as a precautionary measure.

Translational Filter

As stated previously, the state vector for the translational filter is:

$$\underline{x} = \begin{bmatrix} \underline{x}_{plant} \\ \delta\underline{x}_{sensor} \end{bmatrix} \text{ where } \underline{x}_{plant} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

and $\delta\underline{x}_{sensor}$ is a vector of sensor error estimates, which represents the filter's knowledge of the errors in the sensors providing measurements.

The time-varying state dynamics are given by $F = \begin{bmatrix} F_{plant} & 0 \\ 0 & F_{sensor} \end{bmatrix}$, unless no sensor state dynamics are available, in which case $F = F_{plant}$. In these, F_{plant}, which is governed by the Clohessy-Wiltshire equations [2], contains the state dynamics for the plant as follows:

$$F_{plant} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2\omega \\ 0 & -\omega^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3\omega^2 & -2\omega & 0 & 0 \end{bmatrix}$$

ϖ is the orbital angular rate of the spacecraft defined as $\omega = 2 * \pi / P$ where P is the orbital period of the target spacecraft. The continuous-time state dynamics matrix F is then converted to the discrete state transition matrix Φ_k.

For the Laser Range and the Radio Frequency Interrogator the relative measurements of the chaser to the target are Range, Azimuth angle and Elevation angle. The sensor Range, Azimuth angle and Elevation angle have bias, noise and scale factor errors estimated in the Kalman filter. The Laser Range Finder and the Radio Frequency Interrogator have sensor error state vectors $\delta\underline{x}_{sensor}$ to estimate the bias and scale factors on the range, azimuth, and elevation measurements individually. $\delta\underline{x}_{sensor}$ has the following form:

$$\delta\underline{x}_{sensor} = \begin{bmatrix} \delta R_{bias} \\ \delta R_{sf} \\ \delta AZ_{bias} \\ \delta AZ_{sf} \\ \delta EL_{bias} \\ \delta EL_{sf} \end{bmatrix}$$

The measurements from the LRF and RFI are assumed to be of the form:

$$z_1 = Range(1 + \delta R_{sf}) + \delta R_{bias} + v_1 \ \ where\ v_1\ is\ measured\ Rang$$

$$z_2 = Azimuth(1+\delta AZ_{sf}) + \delta AZ_{bias} + v_2 \text{ where } v_2 \text{ is measured Azimuth noise}$$

$$z_3 = Elevation(1+\delta EL_{sf}) + \delta EL_{bias} + v_3 \text{ where } v_3 \text{ is measured Elevation noise}$$

For the extended Kalman filter the measurement equation has to be linearized about the current state estimate, and the mapping matrix H_k and the expected measurements h_k have to be computed for each time step. For the Laser Range Finder and the Radio Frequency Interrogator we have:

$$\underline{h}_k(\hat{\underline{x}}_k^-) = \begin{bmatrix} (1+\delta R_{sf})*R+\delta R_{bias} \\ (1+\delta AZ_{sf})*atan(-Z/X)+\delta AZ_{bias} \\ (1+\delta EL_{sf})*asin(-Y/R)+\delta AZ_{bias} \end{bmatrix} \quad where\ R=\sqrt{(X^2+Y^2+Z^2)}$$

$$(H_{vehicle})_{LRF/RFI} = \begin{bmatrix} \frac{X(1+\delta R_{sf})}{R} & \frac{Y(1+\delta R_{sf})}{R} & \frac{Z(1+\delta R_{sf})}{R} & 0 & 0 & 0 \\ \frac{Z(1+\delta Az_{sf})}{X^2+Z^2} & 0 & \frac{-X(1+\delta Az_{sf})}{X^2+Z^2} & 0 & 0 & 0 \\ \frac{XY(1+\delta EL_{sf})}{R^2\sqrt{X^2+Z^2}} & \frac{-\sqrt{X^2+Z^2}(1+\delta EL_{sf})}{R^2} & \frac{YZ(1+\delta EL_{sf})}{R^2\sqrt{X^2+Z^2}} & 0 & 0 & 0 \end{bmatrix}$$

$$(H_{sensor})_{LRF/RFI} = \begin{bmatrix} 1 & R & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & a\tan(-\frac{Z}{X}) & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & a\sin(-\frac{Y}{R}) \end{bmatrix}$$

The VBGS and the ULTOR sensors provide relative position measurements in the Cartesian LVLH frame instead of providing range, azimuth, and elevation. Each of these measurements has a bias and a scale factor associated with it. The VBGS and the ULTOR sensor error state vector $\delta \underline{x}_{sensor}$ has the form $\delta \underline{x}_{sensor} = [\delta \underline{x}]$ where the elements of $\delta \underline{x}$ are the range error states:

$$\delta \underline{x} = \begin{bmatrix} \delta X_{bias} \\ \delta X_{sf} \\ \delta Y_{bias} \\ \delta Y_{sf} \\ \delta Z_{bias} \\ \delta Z_{sf} \end{bmatrix}$$

The measurements from the VBGS and the ULTOR are assumed to be of the form:

$$z_1 = X(1+\delta X_{sf}) + \delta X_{bias} + v_1 \text{ where } v_1 \text{ is measured XRange noise}$$

$$z_2 = Y(1+\delta Y_{sf}) + \delta Y_{bias} + v_2 \text{ where } v_2 \text{ is measured Y Range noise}$$

$$z_3 = Z(1+\delta Z_{sf}) + \delta Z_{bias} + v_3 \text{ where } v_3 \text{ is measured ZRange noise}$$

For the extended Kalman filter the measurement equation has to be linearized about the current state estimate and the mapping matrix H_k and the mapping function h_k have to be computed for each time step. For the VBGS and the ULTOR we have the following:

$$\underline{h}_k(\hat{\underline{x}}_k^-) = \begin{bmatrix} X(1+\delta X_{sf}) + \delta X_{bias} \\ Y(1+\delta Y_{sf}) + \delta Y_{bias} \\ Z(1+\delta Z_{sf}) + \delta Z_{bias} \end{bmatrix}$$

$$(H_{vehicle})_{VBGS/ULTOR} = \begin{bmatrix} (1+X_{sf}) & 0 & 0 & 0 & 0 & 0 \\ 0 & (1+Y_{sf}) & 0 & 0 & 0 & 0 \\ 0 & 0 & (1+Z_{sf}) & 0 & 0 & 0 \end{bmatrix}$$

$$(H_{sensor})_{VBGS/ULTOR} = \begin{bmatrix} 1 & R_x & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & R_y & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_z \end{bmatrix}$$

Rotational Filter

The rotational extended Kalman filter uses Euler angles and Euler angle rates to estimate the attitude parameters. The state vector for this filter has 24 elements. The first six elements of the state vector are the relative Euler angles and angular rates. The last 18 states are the sensor error state estimates. There are six error states per sensor, with each channel (yaw, pitch, or roll) having both a bias and a scale factor.

$$x = \begin{bmatrix} \underline{x}_{plant} \\ \delta \underline{x}_{sensor} \end{bmatrix} \text{where } \underline{x}_{plant} = \begin{bmatrix} roll \\ pitch \\ yaw \\ roll\ rate \\ pitch\ rate \\ yaw\ rate \end{bmatrix}$$

and $\delta \underline{x}_{sensor}$ is a vector containing six error states for each of three sensors, one of which is shown here:

$$\delta \underline{x}_{sensor} = \begin{bmatrix} roll\ bias \\ roll\ scale\ factor \\ pitch\ bias \\ pitch\ scale\ factor \\ yaw\ bias \\ yaw\ scale\ factor \end{bmatrix}$$

The continuous-time state dynamics matrix for the system is a 24x24 given as follows:

$$F = \begin{bmatrix} F_{plant} & 0 \\ 0 & F_{sensor} \end{bmatrix}$$

where $F_{plant} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

and F_{sensor}, the state dynamics matrix for the sensor error states, is an empty 18x18 matrix.

Similarly, the B matrix is a 24x3 given as follows:

$$B = \begin{bmatrix} B_{Plant} \\ B_{Sensor} \end{bmatrix}$$

where $B_{plant} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

and B_{Sensor} is an empty 18x3 matrix.

The discrete versions of these matrices used in the state update are created using **Error! Reference source not found.** and **Error! Reference source not found.** above.

The system noise matrices (Q1 and Q2) are shown below. The system noise matrix for the sensor error estimates is an empty 18x18 matrix. These are concatenated as shown below to give the entire Q matrix.

$$Q = \begin{bmatrix} Q_1 & 0 & & 0 \\ 0 & Q_2 & & \\ 0 & & \ddots & \\ 0 & & & 0 \end{bmatrix}$$

where Q1 is related to the process noise caused by uncertainty in the plant dynamics and Q2 is the corresponding process noise in the sensor error parameters.

Like the translational EKF, each sensor has its own series of 3x6 matrices that compose its mapping matrix; however, unlike the translational EKF, the rotational EKF only receives measurements from three sensors, so its mapping matrix is a 3x24 with an $H_{vehicle}$ and 3 $H_{sensors}$ as follows:

$$H_k = [H_{vehicle} \quad (H_{sensor})_1 \quad (H_{sensor})_2 \quad (H_{sensor})_3]$$

With

$H_{vehicle} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$ and

$H_{sensor} = \begin{bmatrix} 1 & R & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & R & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R \end{bmatrix}$ where R is the estimated range to target.

COVARIANCE ANALYSIS

The goals of the covariance analysis were to verify that the filter was performing as expected and that the state estimate values were at or below the anticipated levels. In order to do this, SPARTAN was used to simulate spacecraft trajectories and sensor measurements to feed into the filters. This was done in an open-loop fashion with the filter outputs not feeding into the guidance and control algorithms.

The trajectory used in the docking scenario used for this analysis starts with the chaser vehicle 600 meters below the target vehicle, gaining on it at roughly 1 m/s. After it passes beneath the target vehicle, the chaser executes a burn to arrive at a point in front of the target vehicle on its V-bar with a specified, nonzero velocity. After arriving at this point, the relative velocity is damped out and then a glideslope approach is initiated. This can be seen in the figure below.

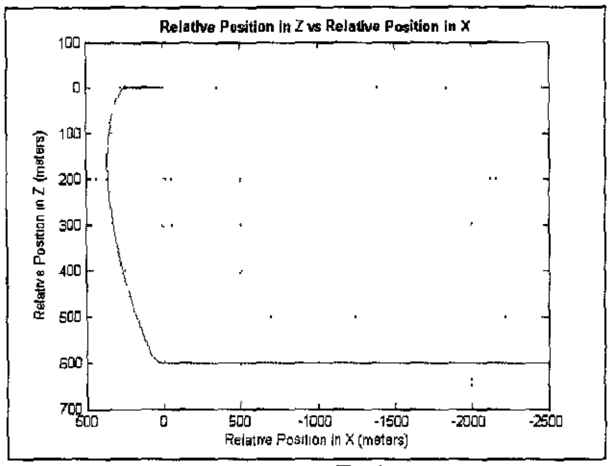

Figure 1: Relative Trajectory Plot.

The figure below shows the relative position estimate errors and uncertainties, as well as the range errors and uncertainties. Note that as the scenario progresses and the distance between the chaser and target vehicles is reduced (not shown) that the uncertainty and noise are decreased. At certain distances shorter-range sensors are brought into range and the uncertainty is reduced drastically. Also note how around 2500 seconds into the scenario the uncertainty is switching from the Z axis (R-bar) into the X axis (V-bar). This is because the actual error ellipsoid that the long-range sensors create is lens-shaped, and that lens rotates through almost 180 degrees between 2000 and 3500 seconds. This encompasses the time prior to and during the thrust onto the V-Bar. Also note that the errors were within the 1-sigma uncertainties shown on the set of axes below the axes with errors.

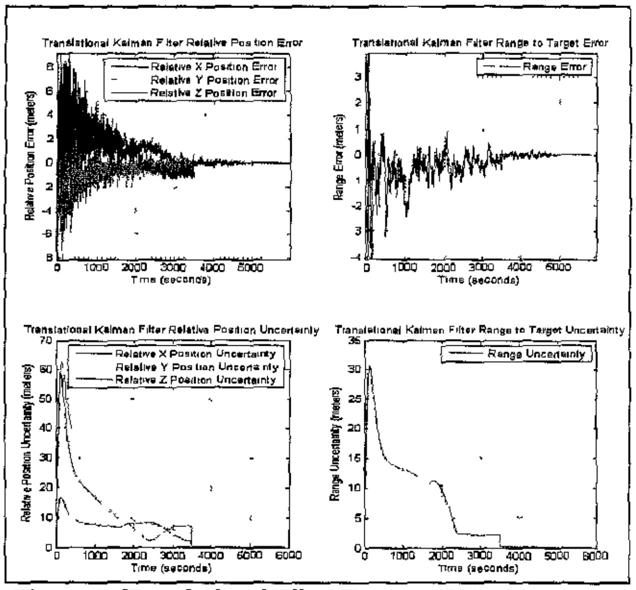

Figure 2: Translational Filter Error and Uncertainty as a function of time.

The figure below shows the position errors and uncertainties during the terminal phases of docking. The spikes in the uncertainty are due to a measurement failing a statistical acceptance test (described later) and thus not being incorporated. The Z-axis position estimate has more error than is anticipated by the uncertainty estimate. This is because the actual sensor noise parameters were set according to a Gaussian distribution, and on this particular run that noise parameter for that terminal docking sensor was at a 2.5 sigma value. The range does not go all the way to zero because the range shown is not the range until contact occurs, but the range between the sensor and its target, and as such cannot go completely to zero.

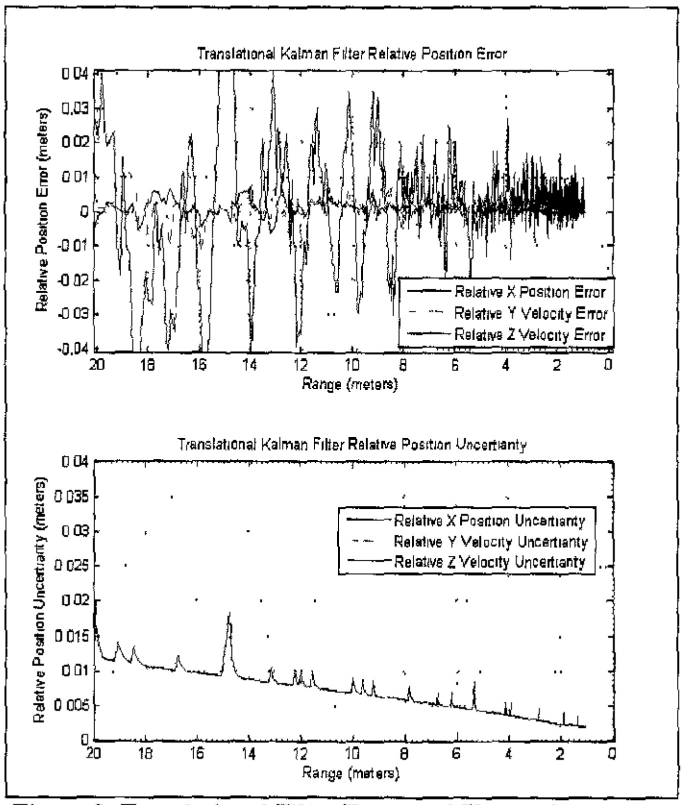

Figure 3: Translational Filter Error and Uncertainty as a function of range.

The filter was also able to converge on the bias and scale factor parameters for the various sensors fairly well using the orbital dynamics embedded in the filter and using measurements from other sensors. In general, the filter was able to converge on the bias parameters better than the scale factor parameters. Below is a figure showing the simulated Laser Range Finder biases and the filter's estimates of these biases.

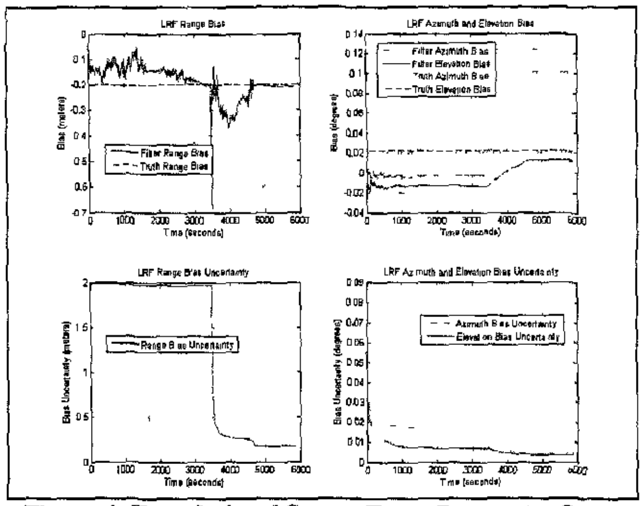

Figure 4: Translational Sensor Error Parameter State Estimation.

The relative attitude extended Kalman filter performed as expected with regard to estimating the relative attitude state and estimating the sensor biases of the short-range sensors. Below is a plot of the attitude errors of the terminal docking phase and the related uncertainties. Although the errors are not actually less than the uncertainty, this is because the bias errors on the sensors are quite large. This filter uses linear attitude equations because it was intended for use in terminal docking only, where relative angles are near zero and relative angular rates are very near zero. Hence, there is no dynamics that would be of use to help the filter estimate these biases. In order to estimate the errors on a sensor, the filter essentially waits for the next more accurate sensor to turn on.

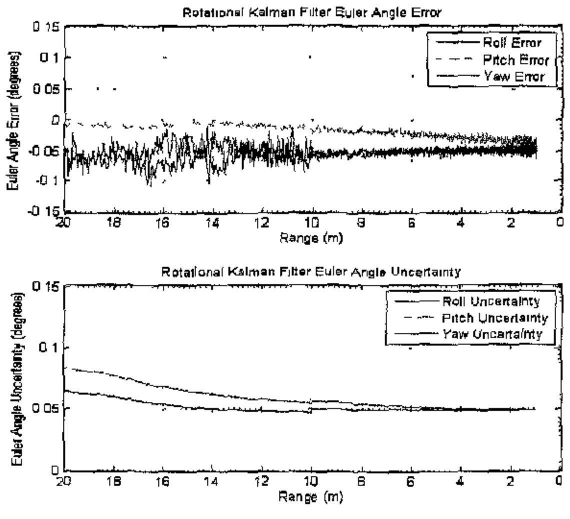

Figure 5: Rotational Filter Error and Uncertainty as a function of range.

Below are shown the simulated bias and the attitude filter's bias estimates for the ULTOR visible sensor.

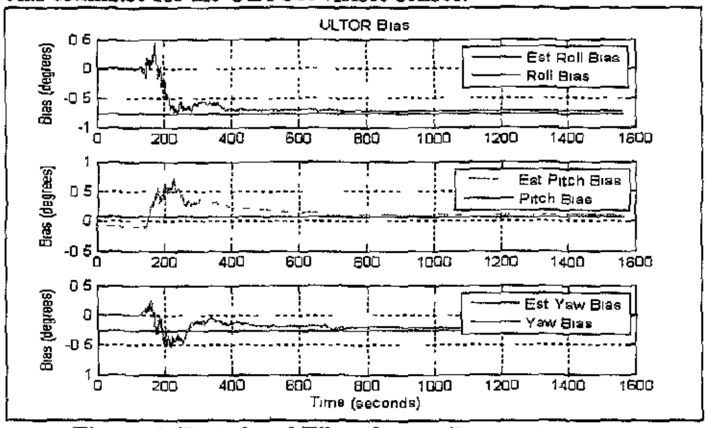

Figure 6: Rotational Filter Sensor Parameter Estimation.

CLOSED-LOOP PERFORMANCE AND DISPERSION EVALUATION

The filters were tested in closed-loop fashion with the rest of the guidance and control algorithms to determine their stability and identify any negative effects. Dispersion analysis is a critical tool in system development, since it provides the ability to assess system performance under random variation of system parameters and external influences. This information is crucial in determining bounds of operation for requirements analysis, for quantifying the relative risks in the system development, and in modeling system uncertainty for robust stability analysis. To perform this analysis, SPARTAN was augmented with Monte Carlo capability.

It is desired that the dispersion values in this analysis be set as close as possible to the 3σ values of the variables' distribution. The variables were grouped according to 12 different categories: target initial conditions, target mass properties, chase vehicle initial conditions, chase vehicle mass properties, main engine parameters, RCS parameters, LRF sensor errors, RFI sensor errors, VBGS far sensor errors, VBGS near sensor errors, ULTOR sensor errors, and chase vehicle controller parameters.

The Monte Carlo analysis was conducted for 520 dispersion cases. The mission case simulated was CEV docking with ISS. Using the parallel processing capability, all of these simulations were conducted within an 8 hour period. 97% of the 520 simulations completed the AR&D mission successfully. Of the simulations that failed, none of the failures were due to problems related to relative navigation software. The carpet plot of successful runs is given in Figure 7.

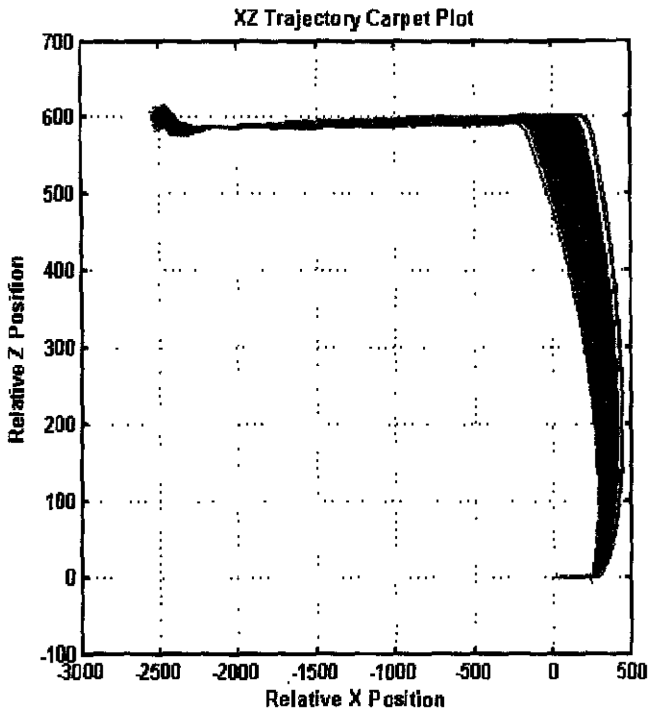

Figure 7: Monte Carlo Trajectory Dispersion Plot

One of the goals of this navigation filter development project was to end up with filters that were credible for use on the actual Constellation vehicles. One of the criteria for that is that the filters must be able to run for extended periods of time without experiencing numerical stability problems. To this end, a Bierman filter has been implemented as an option due to its long term numerical stability. No numerical stability problems were encountered in the Monte Carlo testing, possibly due in part to the length of the simulation runs. Longer duration runs are planned where numerical conditioning problems may become more evident.

The Bierman (U-D) factorization algorithm is a variant of the square-root filter and utilizes a decomposition of the P matrix into a unit upper-triangular matrix and a diagonal matrix, as the propagation of U and D factors is better numerically conditioned than traditional Kalman implementations. Using a vector measurement, the noise covariance matrix R becomes diagonal. This method, despite being more complicated to implement and taking a larger program memory, is known for its fast output. Future work will look at the comparative advantage of Bierman against the traditional methods already utilized.

FILTER PERFORMANCE AS EMBEDDED REAL-TIME ALGORITHMS

One of the goals of this navigation filter development project was to end up with filters that were credible for use on the actual Constellation vehicles. One of the criteria for that is that the filters cannot consume an unreasonable amount of computing resources. In order to evaluate these filters with respect to this criteria, the filters were converted from Simulink and Embedded M-files to C code using MatLab's Real Time Workshop. The autocoded filters were then compiled for execution in a VxWorks environment on a IBM PPC 750 GX processor.

The number of filter states was reduced for this part of the project because the autocoded filters were candidates for later use on another embedded system that was resource-limited. Reducing the number of states was feasible without altering the execution or structure of the filter because there were only two sensors scheduled for use in this testing, instead of five. The number of states was thus reduced from 36 to 18 for each of the translational and rotational filters, dropping six states per absent sensor.

The compiled filters were then fed with artificially generated measurements to allow them to run nominally. The timing analysis indicated that for both of the filters to execute one time took 0.0041 seconds. This represents a very reasonable amount of resource consumption. Also, the translational filter may not need to be run at as fast a rate as the attitude filter, thereby artificially reducing resource consumption.

REAL-TIME TESTING IN THE FRL

Testing occurred in the Flight Robotics Lab on Monday, August 27, 2007 through Friday, August 31, 2007. The FRL's Dynamic Overhead Target Simulator (DOTS) is an overhead, eight degree-of-freedom gantry in the FRL used to position the sensor targets. The movements of the arm correlate to a simulated trajectory flown by the target vehicle in an autonomous rendezvous and docking maneuver. The gantry is equipped with encoders on each axis to provide the true position and orientation of the sensors. The testing used an Advanced Video Guidance Sensor Block II and an ULTOR passive optical correlator sensor system, both developed by Advanced Optical Systems.

bd Systems implemented Extended Kalman filter algorithms to combine the measurements from the multiple HYDRA sensors into a single, reduced-noise, optimal estimate of the relative state of the sensors and docking targets. The Extended Kalman filter algorithms used during testing were a modified version of the filters in SPARTAN, the simulation package for autonomous rendezvous test and analysis developed by bd Systems. SPARTAN is a high-fidelity, on orbit simulation tracking multiple 6 degree-of-freedom vehicles designed to support NASA's new vision for space exploration. SPARTAN is used to test AR&D sensors and guidance, navigation, and control (GN&C) algorithms in a closed-loop fashion.

The Extended Kalman filter used for testing reduced the number of input sensors from the SPARTAN version with five sensors to two sensors—the AVGS and ULTOR. Since the trajectories to be flown in the FRL were not necessarily based on orbital mechanics, the filters were further simplified by eliminating the modeling of orbital dynamics in the filter state transition matrix. The Simulink block diagram for the filter was then converted into C code utilizing the Mathworks Real-Time workshop. This C code was built into a library that was called repeatedly by bdStudio throughout testing.

One version of the C implementation of the Kalman filters was compiled, run, and tested on an embedded computer (PowerPC 750) running VxWorks as the real-time operating system. The executed code on VxWorks matched the outputs from the code executed on a laptop running Windows and the execution speed was acceptable.

Measurement Acceptance Testing

One part of making the filters credible for use in a space-flight environment is making the filter robust to very erroneous or scrambled/meaningless measurements. A Kalman filter that does not do any measurement acceptance testing can have its state estimate greatly degraded by a single corrupted measurement or by a handful of very bad measurements. Excessively noisy measurements can also degrade the filter's performance by causing the estimate to jump around more than it should. The filter can take a long time to recover the estimate to its previous accuracy. Testing each measurement for its likelihood of being a valid measurement is therefore beneficial. The standard method for performing this is through analysis of each measurement's Chi-Squared statistic. The statistic is calculated by

$$\chi^2 = \frac{\varepsilon_k^T * [H_k P_k^- H_k^T + R_k]^{-1} * \varepsilon_k}{m}$$

where the innovation ε_k^T is given in **Error! Reference source not found.** and should be interpreted as the difference between the actual measurements and the expected measurements. Note that m is the length of the measurement vector for that particular measurement. For the sensors described in this report, m = 3 for all of the sensors.

After the Chi-Squared statistic is calculated for a measurement, it is then compared to the then-current *maximum allowable* Chi-Squared value. If the calculated Chi-Squared value is greater than the maximum allowable value, the measurement is not used to update the filter's state estimate.

If the sensor's error properties are assumed Gaussian, the Chi-Squared test can theoretically be expected to cull a predictable percentage of the regular measurements that simply have larger noise components; however, in practice, the Chi-Squared distribution of the actual measurements rarely matches the theoretical distribution. Therefore, the maximum allowable Chi-Squared value must be set through observation of the filter in operation.

It can be seen in Figure 8 below that a large number of the measurements had Chi-Squared values greater than the maximum allowed value. This was caused by the ULTOR being assigned a much larger noise parameter than was reported to the filter through the R matrix. Theoretically, the maximum allowed Chi-Squared value of 6 should reject just under 1 % of the measurements if the measurement errors are unbiased and have the same variance as is reported in that sensor's R matrix. The figure shows that the LRF's statistical behavior is close to expected prior to the ULTOR becoming active. It is then evident that the ULTOR measurements are more noisy than anticipated. In this implementation of the Chi-Squared measurement test, the measurements whose Chi-Squared values are above the threshold of six would not have been used to update the state estimate.

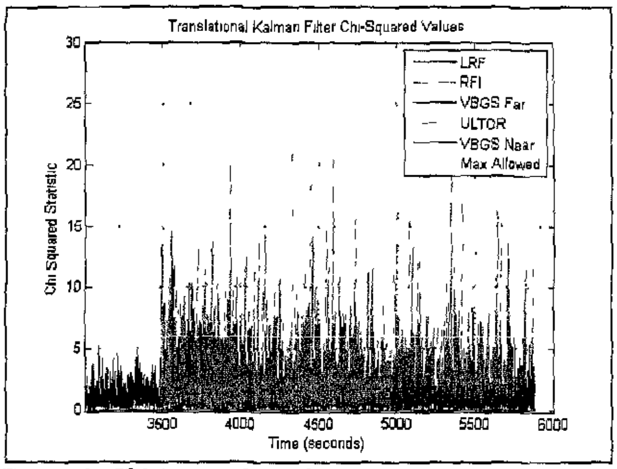

Figure 8: Chi-Squared test for measurement acceptance

Recovery

Under certain circumstances, a Kalman filter's state estimate can be led off course by a series of bad sensor measurements. It is important that the filter be able to recover once good measurements are reintroduced. Unfortunately, the Chi-Squared based measurement testing acts to make the filter reject the new good measurements because the filter has converged on the measurements of the bad sensor. When the malfunctioning sensor is turned off, the filter must realize that it is no longer receiving measurements that agree with its estimate and make efforts to re-converge.

This is implemented by having the filter track how many recent filter iterations have resulted in all available measurements failing the Chi-Squared test. If a sufficient number of iterations have had all bad measurements, then the maximum allowable Chi-Squared value is increased. If the filter keeps receiving only more measurements that end up being rejected, the maximum allowable Chi-Squared value is increased further. If this gradual increasing of the maximum allowable Chi-Squared value fails to let the filter converge after set number of iterations, the filter raises the maximum-allowable Chi-Squared value back to initialization levels, while maintaining the current state estimate and state covariance matrix.

An example of this can be seen in Figure 9 below. It should be noted that the filter had to be configured extremely poorly to cause this degree of divergence. Nevertheless, the ability to re-converge is seen at 3500 seconds and at 4900 seconds. Around 5100 seconds, the filter experiences enough bad measurements that it starts to re-initialize. Unfortunately, the poor filter configuration that had to be used to test this filter feature does not allow the filter to re-converge after the re-initialization, and a series of re-initializations are then executed in rapid succession.

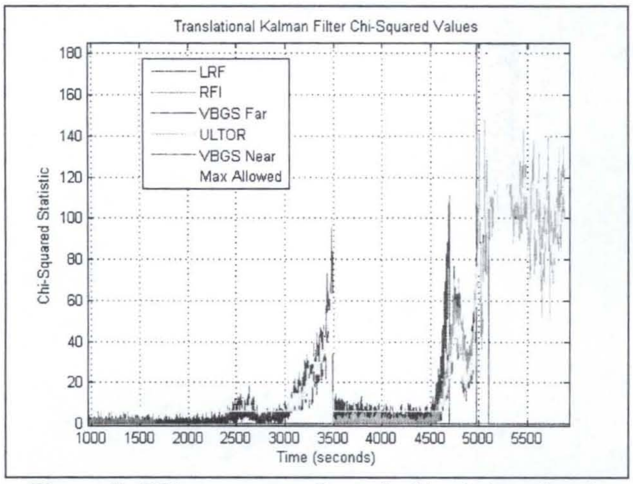

Figure 9: Filter recovery from faulty state estimate

Measurement Drop-outs

The Kalman filter implementation developed for this application is robust to two types of measurement drop-outs, and which of these is more relevant depends on the implementation in the code controlling the system. The first type of drop-out is described as a sensor measurement sitting in a buffer waiting to be read by the filter until it is replaced by a new sensor measurement. The problem is that the filter may execute several times before the sensor measurement is updated. The filter is robust to this by not allowing the same measurement to be used to update the state estimate more than once. The second type of measurement drop-out is described as the sensor measurement in the buffer being replaced by zeros when the filter reads the measurement or when no measurement is available. A measurement of this type is also detected by the filter as invalid and is not used to update the state estimate. This makes it simple to communicate to the filter that a sensor has been turned off; zeros can be written to the measurement buffer or the previous measurement can be left in the buffer to wait for an updated measurement.

bdStudio is an internally developed environment for visual representation of models and simulations. It has the ability to load 3D models from commonly used CAD applications. In addition to loading 3D models, bdStudio can add 3D visual effects caused by the lighting from the sun and can load a mapping of the stars. The position and orientation of the 3D models in bdStudio are modified by Python scripts. The power of Python scripting enables various applications to control the behavior of the models, such as Matlab/Simulink, TRICK, MAVERIC, or other simulation programs. A sample view of bdStudio can be seen in Figure 10.

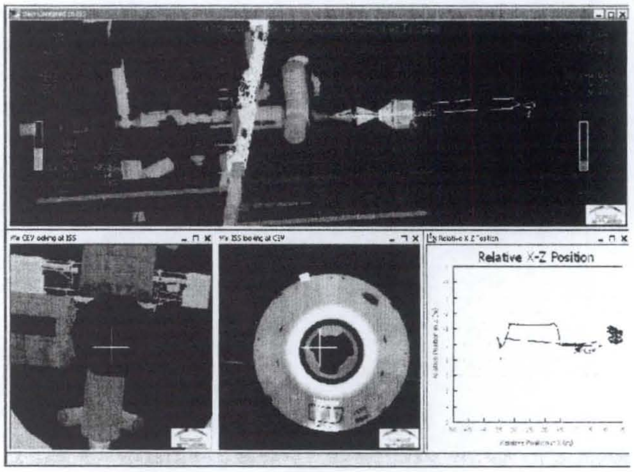

Figure 10: bdStudio Screenshot

Test Process

The DOTS gantry was used to maneuver the AVGS and ULTOR sensors along predefined trajectories relevant to AR&D missions. The encoder data from the gantry provided the truth data regarding the positioning of the sensors. The data generated by the two AOS sensors was transmitted to bdStudio to graphically display and pass through the Extended Kalman filters in real-time. All data was recorded to have available for further post-processing analysis.

AOS transmitted raw sensor data over a UDP Ethernet connection and a Python script had been written to listen for data on UDP port 6768. Upon receiving the data, the script would pass the raw sensor information off to the extended Kalman filter and update the graphical representation accordingly.

Preliminary Test Results

A preliminary analysis of the testing results showed the testing accomplished its primary objectives, and some insight was gained into the areas to focus on in future work. The Extended Kalman filters performed well. The filters had been tuned to values that were correct for use in SPARTAN, the autonomous rendezvous and docking simulation developed by bd Systems. This tuning includes the initial state covariance matrix, the system covariance matrix, and the measurement covariance matrices. These tuning parameters, however, were not highly accurate for the FRL testing for several reasons. The measurement covariance matrices were set *a priori* based on error characteristic estimates provided by AOS before any testing had commenced. Hence the certainty in those values was low. Additionally, the system covariance matrix was based on SPARTAN simulations in which the main perturbations were caused by thruster firings that decreased in magnitude as the chaser vehicle approached the target vehicle.

A preliminary analysis reveals the filters reduced the noise from the sensors. An example of the effectiveness of the filters in smoothing the data from the AVGS alone is displayed in Figure 11. The data in this figure was recorded during a trajectory simulation of the DOTS in which the gantry moved the sensor targets towards the sensors. Note that truth data from the DOTS was not available in time for this quick-look analysis of filter performance; this data will of course provide the ultimate measurement of the improvements provided by the Kalman filter algorithms.

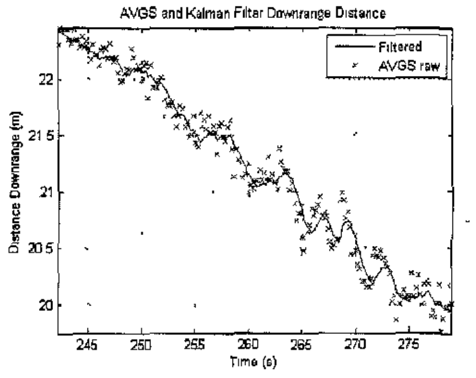

Figure 11: Downrange distance vs. time

The data sampling rate for the filters proved sufficient such that acceleration of the sensor targets did not cause a response lag by the filter. As seen in Figure 12, the red line representing the filtered data responded nearly simultaneously as the change in the raw data.

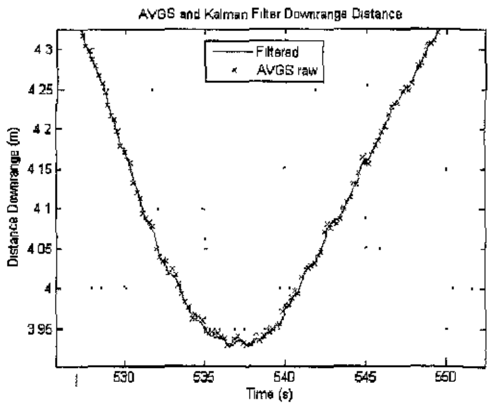

Figure 12: Downrange distance vs. time during acceleration of DOTS

The filter also successfully combined the measurements from the AVGS and ULTOR, as seen in Figure 13. The filtered data will be more accurate after further examination of the sensor noise characteristics and the covariance matrices are tuned.

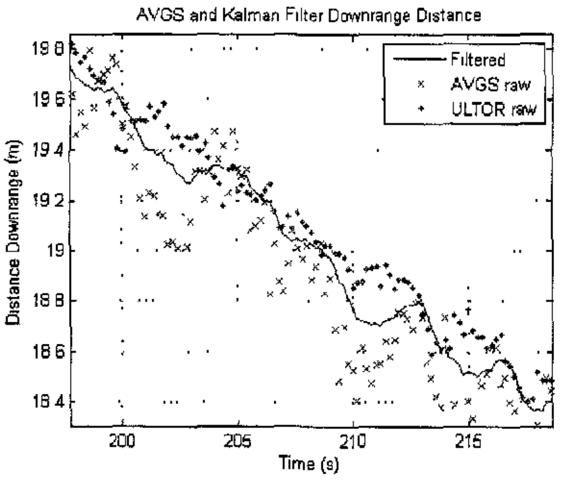

Figure 13: AVGS and ULTOR data combined by the Kalman filter

As mentioned earlier, the noise characteristics of the sensors were not well known when the filters were tuned before testing. This lack of knowledge regarding the noise characteristics was apparent for the AVGS, which exhibited larger attitude errors than previously anticipated. These large errors cause two effects: the first result is the suboptimality of the filter since the estimate of the noise differs significantly from the actual noise. The second effect is the rejection of valid sensor data as a consequence of the Chi-Squared testing being performed on the measurement testing. Since the AVGS' attitude estimates were more noisy than anticipated, a much larger-than-anticipated portion of the measurements were rejected. This data rejection resulted in less confidence in the estimates of the attitude filter and occasionally losing track of attitude information. Because the filtering algorithm reduces data selectivity when the frequency of measurement rejection increases, the filters do re-acquire a tracking of attitude information. It was observed that the current filtering algorithms are tuned to re-acquire too slowly once lost. This prolonged re-acquire time will be addressed in future updates to SPARTAN.

Another problem encountered during testing was error accumulation (about 20 degrees in 15 minutes) in the yaw axis of the attitude Extended Kalman filter due to its error states. This inaccuracy only occurred during normal operation when attitude measurements were accepted from the AVGS alone. The AVGS yaw scale factor error state in the Kalman filter was being excited. This scale factor is associated with distance, not the angle. After a preliminary investigation, it appears that this error growth is caused by the highly correlated noise produced by the AVGS.

An example of this correlated noise is shown in Figure 14 below. This figure contains approximately 2000 data points that were collected consecutively while the DOTS was stationary. The correlation is less evident in the Yaw-Z axes than other axes, like the Pitch-Z axes; yet the Yaw-Z coupling is believed to cause the observed error accumulation due to the structure of the filter. As noted previously, this problem occurred only when the ULTOR was producing unacceptable attitude measurements. A temporary solution for this problem was devised and implemented. This solution involved locking the attitude scale factor errors at zero, whereas the handling of the bias errors was unchanged as they compensate for any difference in alignment of the sensors or docking targets. This change was implemented near the end of the testing period, and the underlying conditions that caused the problem were not encountered again (ULTOR not functioning).

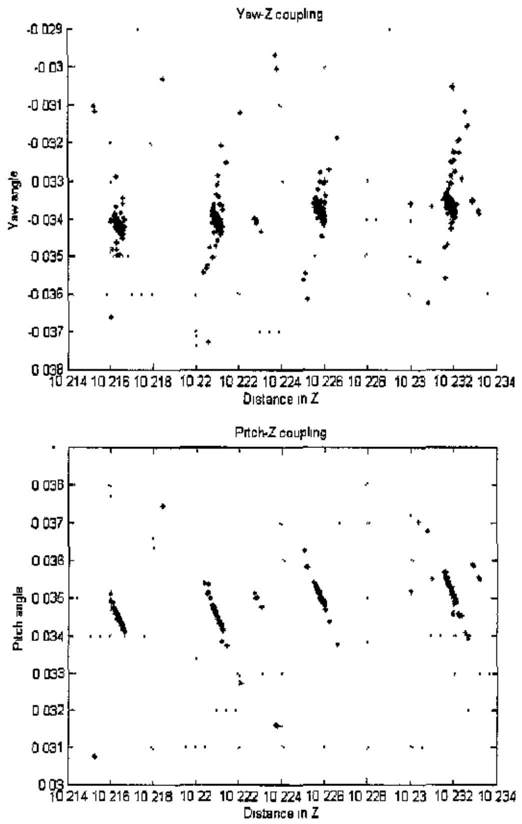

Figure 14: Correlated noise from the AVGS

ACKNOWLEDGMENTS

The authors would like to thank Dr. Connie Carrington of Marshall Space Flight Center for sponsoring the work that led to this paper. They would also like to acknowledge the staff of the FRL for helping with the real time testing and data collection, as well as Stan Klinzak for his part in the VxWorks testing and David Stark for some of his support on the preliminary filter development.

REFERENCES

[1] Cruzen, C. A., Lomas, J. L., and Dabney, R. W., "Test Results for the Automated Rendezvous and Capture System," AAS 00-003, *AAS Guidance and Control Conference*, Feb. 2000, pp. 4-5.

[2] Fehse, W., *Automated Rendezvous and Docking of Spacecraft*, Cambridge Aerospace Series, Cambridge University Press, Cambridge, UK, 2003.

[3] Stevens, B. L. and Lewis, F. L., *Aircraft Control and Simulation*, 2nd ed., Wiley, Hoboken, NJ, 2003, pp. 33-47.

[4] Lemoine, F. G. R., Smith, D. E., Zuber, M. T., Neumann, G. A., and Rowlands, D. D., "A 70th Degree Lunar Gravity Model (GLGM-2) from Clementine and Other Tracking Data," *Journal of Geophysical Research*, Vol. 102, No. E7, 1997, pp. 16339-16359.

[5] Wertz, J. R. (Ed.), *Spacecraft Attitude Determination & Control*, D. Reidel, Dordrecht, Holland, 1978, pp.567.

[6] Lide, D. R. (Ed.), *CRC Handbook of Chemistry and Physics*, 82nd ed., CRC Press, Boca Raton, FL, 2001.

[7] Jacchia, L. G., "Thermospheric Temperature, Density, and Composition: New Models", SAO Special Report No. 375, 1977.

[8] Hablani, H. B., Tapper, M., and Dana-Bashian, D., "Guidance Algorithms for Autonomous Rendezvous of Spacecraft with a Target Vehicle in Circular Orbit," AIAA-2001-4393, *AIAA Guidance, Navigation, and Control Conference*, Aug. 2001, pp. 1- 5.

[9] Wie, B., Weiss, H., and Arapostathis, A., "Quaternion Feedback Regulator for Spacecraft Eigenaxis Rotation," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 12, No. 3, 1989, pp. 375- 380.

[10] Wie, B., *Spacecraft Vehicle Dynamics and Control*," AIAA Education Series, AIAA, Reston, VA, 1998.