JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 37, No. 4, July–August 2014

Computationally Distributed Real-Time Dual Rate Kalman Filter

Stephen Steffes*
DLR, German Aerospace Center, 28359 Bremen, Germany

DOI: 10.2514/1.G000179

Many systems include sensors with large measurement delays that must be fused in a Kalman filter in real time. Often, the filter state must be propagated at a higher rate than the rate at which measurements are taken. This can lead to a significant amount of unused CPU time during the time steps in which no measurements are available. This paper presents a method of fusing delayed measurements for a restricted set of systems, which more efficiently uses processing resources at the expense of data availability. The new method splits the filter into a high-rate and a low-rate task running in parallel. The high-rate task propagates the whole states, and the low-rate task propagates and updates an error state filter, which can be distributed over several high-rate periods.

I. Introduction

THIS paper considers the problem of designing efficient real-time discrete Kalman filters for systems with latent sensor measurements. The focus is on a particular set of systems in which new measurements are not taken until delayed measurements are available, which is a subset of the more general problem of fusing measurements with general frequencies and delay times. This is a common situation in the field of vehicle navigation, in which high-rate (HR) low-latency control data (for example, accelerometer, gyro, wheel encoder, etc.) are used to propagate a filter in time, and low-rate (LR) high-latency observation measurements (ex. GPS, star tracker, vision systems, etc.) are used to update the filter. Often, the designer has the option to synchronize measurements (for example, by scheduling sensor triggers) such that multiple sensors are sampled at the same time. This reduces the frequency of Kalman filter (KF) update steps, making it more likely that the system fits the particular set of systems explored by this work.

Often, the timeline for a real-time implementation of a KF looks like Fig. 1. The KF is propagated at each filter time step k from the previous time t_{k-1} to the current time t_k. Whenever delayed measurements are available at time step m, the KF must additionally be updated at that time step (taking the delay into account). Propagation is done at HR, and updates are done at LR, where the frequency of the LR operations is less than or equal to the frequency of the HR operations. A real-time timing requirement is also imposed, which requires that the operations for each HR period are complete before the next time step occurs, thus ensuring that the KF results can be output to the user synchronously. Therefore, the computer must have enough resources to ensure that during time step m both the propagation and update computations complete within one HR period. Assuming the propagation computations take the same amount of time regardless of the time step, there will always be a period of time between the end of the propagation computations and the beginning of the next time step for every HR time step that does not include an update step (such as all time steps except m in the figure). This can result in a large amount of unused processor time if the LR frequency is much lower than the HR frequency or if there is a large amount of computations per LR measurement (such as when additional processing is needed to convert the measurement into the form required by the KF).

After an extensive literature survey, no work was found that addresses the sensor latency problem and effectively uses this unused CPU time. All real-time Kalman filter implementations use a series of serial propagate and update steps, similar to Fig. 1, and thus generally do not effectively use the CPU. This includes methods that specifically address the problem of fusing latent sensor measurements, such as incorporating delayed states in the measurement equation [1], extrapolating the measurement to the current time [2], augmenting the state vector with delay states [3], recalculating the filter, and other nonoptimal methods [4]. There are several papers that present decentralized (or distributed) Kalman filter architectures [5–7], which spread out computations by implementing parallel processing schemes, but these address the entirely different problem of combining results from multiple parallel filters.

This paper presents a real-time dual rate version of the KF, which not only solves the sensor latency problem but also generally uses the CPU more efficiently. In doing so, a new metric for evaluating algorithm performance is created, which measures the minimum processing power needed to meet the real-time timing requirement. The new dual rate filter is similar to Larsen's measurement extrapolation method [2] but has some key differences. Variations of the dual rate filter using the extended KF have already been implemented in a number of systems showing some success [8,9], including the hybrid navigation system, which flew on the SHEFEX2 sounding rocket mission [10,11].

II. System and Filter Equations

A linear discrete system with additive Gaussian noise can be modeled as

$$\boldsymbol{x}_k = \Phi_{k-1}\boldsymbol{x}_{k-1} + B_{k-1}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{1}$$

$$\mathbf{y}_k = H_k\boldsymbol{x}_k + \nu_k \tag{2}$$

where $\mathbf{w}_k \sim \mathcal{N}(0, Q_k)$ and $\nu_k \sim \mathcal{N}(0, R_k)$. Assuming the two noise vectors are independent, the optimal state estimator is a KF [12].

In this work, it is useful to use the error state formulation of the KF (commonly used for attitude estimation [13]), which is found by estimating the error state vector $\delta\boldsymbol{x}$ rather than the whole state vector $\boldsymbol{x}$. The error state vector is defined as in Eq. (3), giving the system model in Eq. (4). The assumed form of the measurement model is in Eq. (5), which directly observes the error states:

$$\delta\boldsymbol{x}_k \equiv \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k \tag{3}$$

$$\delta\boldsymbol{x}_k = \Phi_{k-1}\delta\boldsymbol{x}_{k-1} + \mathbf{w}_{k-1} \tag{4}$$

$$\mathbf{z}_k = H_k\delta\boldsymbol{x}_k + \nu_k \tag{5}$$

Received 25 July 2013; revision received 25 September 2013; accepted for publication 26 September 2013; published online 8 April 2014.

*Research Engineer, Navigation and Control Systems Department, Robert-Hooke-Str. 7.

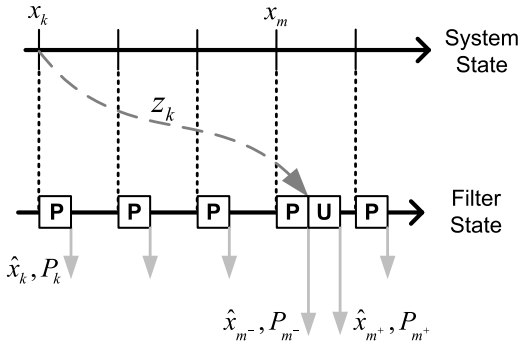

Fig. 1 Serial real-time KF timeline. P blocks are KF propagations, and U blocks are KF updates.

The error state Kalman filter equations are summarized in the following text. Equations (6–8) represent the time propagation step, and Eqs. (9–11) are for the measurement update step:

$$\hat{\boldsymbol{x}}_{k^-} = \Phi_{k-1}\hat{\boldsymbol{x}}_{(k-1)^+} + B_{k-1}\mathbf{u}_{k-1} \tag{6}$$

$$\delta\hat{\boldsymbol{x}}_{k^-} = \Phi_{k-1}\delta\hat{\boldsymbol{x}}_{(k-1)^+} \tag{7}$$

$$P_{k^-} = \Phi_{k-1}P_{(k-1)^+}\Phi_{k-1}^T + Q_{k-1} \tag{8}$$

$$K_k = P_{k^-}H_k^T[H_kP_{k^-}H_k^T + R_k]^{-1} \tag{9}$$

$$\delta\hat{\boldsymbol{x}}_{k^+} = \delta\hat{\boldsymbol{x}}_{k^-} + K_k[\mathbf{z}_k - H_k\delta\hat{\boldsymbol{x}}_{k^-}] \tag{10}$$

$$P_{k^+} = [I - K_kH_k]P_{k^-} \tag{11}$$

$$\hat{\boldsymbol{x}}_{k^+} = \hat{\boldsymbol{x}}_{k^-} + \delta\hat{\boldsymbol{x}}_{k^+}; \qquad \delta\hat{\boldsymbol{x}}_{k^+} \leftarrow 0 \tag{12}$$

Equation (12) is the reset equation, which includes a two-step process used to move information from the error state vector to the whole state vector [13]. The current error state estimate is simultaneously added to the whole state vector and removed from the error state vector, which resets the error state vector to zero. If the reset equation is used after each update, then the error state vector is effectively always zero, and this formulation of the error state KF is mathematically equivalent to the standard KF.

If the KF used to estimate this system is run in real time, then the measurement information is not available to the filter until some time after the measurement was taken, say M time steps later. Assuming the filter is propagated such that the filter reference time (i.e., filter time of validity) is always close to the system reference time (i.e., real time), then at least some of the information needed from the measurement model is not known until some time step m later, where $m = k + M$. The delayed measurement cannot be fused into the filter without some modification of the error state KF.

III. Dual Rate Filter

Section I discusses a number of different methods to fuse delayed measurements, some of which optimally fuse the delayed data and mainly differ in implementation and computational speed. A dual rate filter is described here, which not only optimally fuses delayed measurements but also distributes the computational burden of some of the filter calculations over time.

A. Filter Description

Generally, the state estimate output is much more important for the user than the covariance output. For example, an autonomous vehicle control system typically uses the state estimate from the onboard navigation system and not the covariances. None of the most popular control methods needs state covariances, including PID, linear quadratic Gaussian, H_2, and H_∞. Therefore, the covariance output could be at a lower rate and/or delayed with respect to the state output with little or no disadvantage to the user.

The dual rate filter takes advantage of this by processing all whole state calculations in a HR high-priority task and all error state and covariance calculations in a LR low-priority task since they are only needed to process observation measurements. The HR task propagates the whole states in time and calculates some variables needed for the LR task. The LR task propagates the covariance matrix to the measurement reference times using a LR covariance propagation routine and updates the covariance matrix and error state vector when measurement information is available. The estimated error states are then fed back to the HR task to correct the whole states. For this discussion, it is assumed that the HR and LR tasks run on a single processor.

Three assumptions are needed in order to apply the dual rate filter to the system. These limit the scope of the system but are common in several types of problems. First, measurement reference times are known at time step k (i.e., at time step k, it is known that a measurement is being taken). Second, a delayed measurement is available before the next measurement is taken (i.e., the next measurement reference time occurs after time step m). Finally, covariances are only needed by the user at LR. Without further loss of generality, it is also assumed that measurements occur at a fixed period of N time steps ($M \leq N$) in order to simplify notation.

The equations needed for the dual rate filter are first described, followed by the filter structure and timing.

Most of the filter calculations are done in the LR task using a LR version of the filter propagation equations, which is equivalent to recursively applying Eqs. (7) and (8) to propagate the error state vector and covariance from time step $k - N$ to some time step j,

$$\delta\hat{\boldsymbol{x}}_{j^-} = \Phi_{(k-N)\to j}\delta\hat{\boldsymbol{x}}_{(k-N)^+} \tag{13}$$

$$P_{j^-} = \Phi_{(k-N)\to j}P_{(k-N)^+}\Phi_{(k-N)\to j}^T + Q_{(k-N)\to j} \tag{14}$$

where $\Phi_{(k-N)\to j}$ and $Q_{(k-N)\to j}$ can be used to propagate the filter from time step $k - N$ to some general time step j and are calculated in the HR task using the following recursive equations:

$$\Phi_{(k-N)\to(k-N+1)} \equiv \Phi_{k-N} \tag{15}$$

$$\Phi_{(k-N)\to j} = \Phi_{j-1}\Phi_{(k-N)\to(j-1)} \tag{16}$$

$$Q_{(k-N)\to(k-N+1)} \equiv Q_{k-N} \tag{17}$$

$$Q_{(k-N)\to j} = \Phi_{j-1}Q_{(k-N)\to(j-1)}\Phi_{j-1}^T + Q_{j-1} \tag{18}$$

All of the filter update equations remain unchanged, except for the reset equation. Equation (12) cannot be used immediately after the update, which is completed at time step l ($l = k + L$, where $M \leq L \leq N$), since the whole states and error states will be at two different reference times. Instead, the error states are first propagated to time step l using Eq. (13) before feeding back the error state estimates to the whole states. The reset equation must then be changed to

$$\hat{\boldsymbol{x}}_{l^+} = \hat{\boldsymbol{x}}_{l^-} + \delta\hat{\boldsymbol{x}}_{l^-}; \qquad \delta\hat{\boldsymbol{x}}_{l^-} \leftarrow 0 \tag{19}$$

which maintains the convention that the $-$ and $+$ represent reference times immediately before and after the update, respectively.

Putting all of these equations together, the dual rate filter structure is shown in Fig. 2, where the following steps are done:

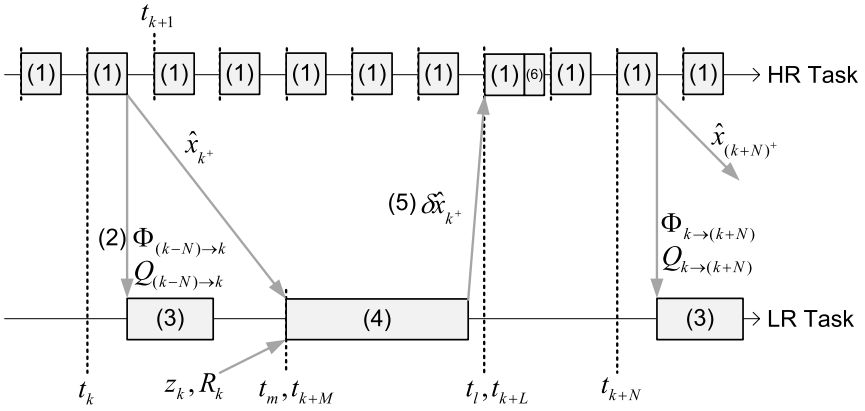

Fig. 2 Timeline of the dual rate filter. See the text for a description of each numbered step.

1) At each time step j, the HR task computes $\hat{\boldsymbol{x}}_{j^-}$ using Eq. (6) and $\Phi_{(k-N)\to j}$ and $Q_{(k-N)\to j}$ using Eqs. (15–18).

2) At each measurement reference time k, the HR task computes and sends $\Phi_{(k-N)\to k}$ and $Q_{(k-N)\to k}$ to the LR task.

3) The LR task then computes P_{k^-} using Eq. (14).

4) When the measurement information is available at time step m, the LR task updates the error state filter using Eqs. (9–11).

5) The LR task then sends $\delta\hat{\boldsymbol{x}}_{k^+}$ to the HR task.

6) During the next time step l, the HR task calculates $\delta\hat{\boldsymbol{x}}_{l^-}$ using Eq. (13) and resets the filter states using Eq. (19).

Note that the HR task has a higher priority than the LR task, so all LR processing is paused any time HR processing occurs. This is needed so that the HR task does not overrun and exceed its real-time timing requirements. It also ensures that the LR task is only running between HR processing blocks.

Using this dual rate filter, the state vector is not updated until time step l. This is generally later than using any of the filtering methods discussed in Sec. I, all of which would update the filter at time step m. The time difference between l and m is determined by the computations needed in the update routines and processing power of the computer. The processing power can be increased such that $l = m$ or decreased such that $l = k + N$ and fit to the filter designer's needs.

B. Comparison to Larsen's Measurement Extrapolation Method

Of all the filtering methods cited in Sec. I, Larsen's measurement extrapolation method [2] is the most similar. This method is similar to the standard KF but uses the following equations to update the filter at time step m when the measurement is available (applied to the particular problem discussed in this work):

$$\hat{\boldsymbol{x}}_{m^+} = \hat{\boldsymbol{x}}_{m^-} + \Phi_{k\to m}K_k[\mathbf{z}_k - H_k\delta\hat{\mathbf{x}}_{k^-}] \tag{20}$$

$$P_{m^+} = P_{m^-} - \Phi_{k\to m}K_kH_kP_{k^-}\Phi_{k\to m}^T \tag{21}$$

Equation (20) is the same as combining Eqs. (10), (13), and (19). Using Eqs. (14) and (21) can be changed to

$$P_{m^+} = \Phi_{k\to m}[I - K_kH_k]P_{k^-}\Phi_{k\to m}^T + Q_{k\to m} \tag{22}$$

which is the same as combining Eqs. (11) and (14).

The main difference is in the dual rate filter's structure and order of equation execution, which brings two key advantages over the measurement extrapolation method and other optimal methods that solve the sensor latency problem. First, distribution of KF processing allows greater usage of the CPU and thus allows less powerful computers to be used. Second, the method is easily adaptable to dual CPU computers since it is composed of two parallel processes.

C. Algorithm Resources

The dual rate filter requires a larger number of calculations compared to the standard KF. There are additional one-time (per update) calculations needed for the error state propagation and reset equations, but the majority of additional calculations come from the covariance propagation routine. Calculating $\Phi_{(k-N)\to k}$ and $Q_{(k-N)\to k}$ requires one more $n \times n \times n$ matrix multiplication per HR time step (where n is the length of the state vector) compared to the standard KF covariance propagation calculations. This is similar to the amount of calculations needed for Larsen's measurement extrapolation method.

However, a more important metric for multithreaded processes is not the number of calculations required but the amount of processing resources needed to meet the real-time timing requirements (i.e., needed processing power). Of course, this depends on the particular system under analysis. When the difference between HR and LR frequency is large or a significant amount of computations is required for the update routine, then the processing power needed for the dual rate filter may be significantly less than for Larsen's method.

To show the importance of this metric, one possible example is explained, as shown in Fig. 3. In this case, $N = 10$, $M = 4$, and $L = 8$. Larsen's measurement extrapolation method runs on computer A, and the dual rate filter runs on computer B. The processing power of computer A is just enough to compute the KF propagation and update steps in one HR period. The processing power of computer B is scaled such that LR propagation (step 3 of the dual rate filter) finishes before time step m and the LR update (step 4) finishes before time step l (where LR processes run in between HR processes). Since the number of computations needed for the

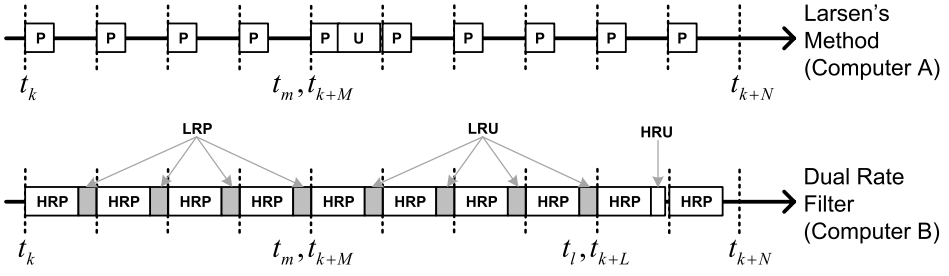

Fig. 3 Example of Larsen's method and dual rate filter timelines. The legend is as follows: propagate (P), update (U), HR propagate (HRP), HR update (HRU), LR propagate (LRP), and LR update (LRU).

propagate and update steps in Larsen's method is roughly the same as for the dual rate filter, it is clear that computer B needs less processing power. This example shows how the dual rate filter running on a computer with reduced processing power can be used to give the same state estimate as Larsen's measurement extrapolation method running on a more powerful computer, although at a later time step (Larsen's method provides an updated state estimate at time step m).

In terms of memory usage, the dual rate filter and Larsen's method both use an equal amount of memory. However, the dual rate filter may have additional CPU and memory requirements that come from the overhead of switching between HR and LR tasks. Although, this is usually minor.

D. Additional Modifications

There are some additional modifications to the dual rate filter that can be employed, which can shift the computational burden to other tasks and/or reduce the total burden.

If any calculations for the update routine can be done before the measurement data are available (for example, calculating H, R, and K), then this can be done in between the LR propagate and update steps (between steps 3 and 4 in Fig. 2), which will shorten the time needed for the LR update, bringing time step l closer to m.

Often Q is small compared to P, making it possible to approximate $Q_{(k-N)\to k}$ with little change to the filter results. This can result in a large computational savings, but it makes the filter nonoptimal. Depending on the system and performance requirements, this could be approximated by summing the process noise over the LR interval [Q' as in Eq. (23)] which was used with some success by [10] or using some other lower-fidelity integration approach, like something akin to Euler integration [Q'' as in Eq. (24)]. For example, using Q' would save two $n \times n \times n$ matrix multiplications per HR interval:

$$Q'_{(k-N)\to k} = \sum_{i=k-N}^{k-1} Q_i \tag{23}$$

$$Q''_{(k-N)\to k} = \sum_{i=k-N/2}^{k-1} Q_i + \Phi_{(k-N)\to k}\left(\sum_{i=k-N}^{k-N/2-1} Q_i\right)\Phi^T_{(k-N)\to k} \tag{24}$$

IV. Example

Consider a general system of the form described in Eqs. (1) and (2), where the lengths of $\boldsymbol{x}$, $\mathbf{u}$, and $\mathbf{y}$ are 9, 3, and 3, respectively, and $N = 10$, $M = 4$, and $L = 8$. The following filters are used to estimate the state vector and account for delayed measurements:

1) Recalculate the filter at m.
2) Use Larsen's measurement extrapolation method.
3) Use the dual rate filter.

Since each of these filters optimally fuses measurements, they all give the same value for $\hat{\boldsymbol{x}}_j$ when $j \notin [m, l-1]$ and for P_{k^-}.

The standard Kalman filter (with no measurement delays) and filters 1, 2, and 3 were implemented as a high-priority process on a CMA157886 CPU module from RTD running the real-time operating system QNX. The propagation and update protions of each algorithm were timed using an accurate real-time clock. The values listed in Tables 1–3 are normalized by the total time needed for the propagate and update steps of the standard KF. Since LR operations for filter 3 are spread out over several time steps, the computational burden shown for the LR propagate and update steps is the sum total over one LR period.

Table 1 Normalized computational burden for filter 1

Time step	Propagate	Update
m	0.45	2.36
Other	0.45	0

Table 2 Normalized computational burden for filter 2

Time step	Propagate	Update
$[k+2, m-1]$	0.66	0
m	0.66	0.81
Other	0.45	0

Table 3 Normalized computational burden for filter 3

Time step	HR propagate	HR update	LR propagate	LR update
$k+1$	0.04	0	— —	— —
l	0.63	0.03	— —	— —
Other	0.63	0	— —	— —
One LR period	— —	— —	0.41	0.56

Table 4 Normalized processing power for selected cases

N	M	L	1	2	3
10	4	8	2.81	1.47	0.78
10	2	8	1.91	1.47	0.73
10	2	3	1.91	1.47	1.20
100	10	20	5.45	1.47	0.69
100	10	80	5.45	1.47	0.64

The normalized needed processing power (as discussed in Sec. III.C) for several select values of N, M, and L are listed in Table 4. This number is equal to the maximum time needed in any HR period scaled such that the processing power of the standard KF is 1. This number can be thought of as the relative CPU speed needed in order to meet the real-time timing requirements (of course, processing power is not just determined by CPU speed). These results suggest that the dual rate filter may need a computer with significantly less processing power than the other methods, which (for any applicable system) largely depends on the value of $L - M$.

V. Conclusions

This work presents a new dual rate method of optimally fusing delayed measurements using a Kalman filter split into two execution threads of variable frequencies. After the filter is updated and before the next measurement is taken, the dual rate filter provides the same results as the standard KF without measurement delays and is thus optimal during these times. The dual rate nature of the method distributes several filter calculations over time, using processing time that is generally unused in real-time systems, allowing the use of computers with lower processing power than what is needed for most other methods. It also allows the use of a low-rate process noise covariance matrix, which may bring significant computational savings. This method comes at the expense of providing covariance information at a low rate and restricting the method to systems in which delayed measurements are received before new measurements are taken.

References

[1] Brown, R., and Hartman, G., "Kalman Filter with Delayed States as Observables," *Proceedings of the National Electronics Conference*, Vol. 24, 1968, pp. 67–72.

[2] Larsen, T., Andersen, N., Ravn, O., and Poulsen, N., "Incorporation of Time Delayed Measurements in a Discrete-Time Kalman Filter," *Proceedings of the 37th IEEE Conference on Decision and Control*, Vol. 4, Inst. of Electrical and Electronics Engineers, 1998, pp. 3972–3977.

[3] Bayard, D., and Brugarolas, P., "An Estimation Algorithm for Vision-Based Exploration of Small Bodies in Space," *Proceedings of the 2005 American Control Conference*, Inst. of Electrical and Electronics Engineers, 2005, pp. 4589–4595.

[4] Der Merwe, R. V., Wan, E. A., and Julier, S. J., "Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor-Fusion: Applications to

Integrated Navigation," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, AIAA, Reston, VA, 2004.

[5] Felter, S., "An Overview of Decentralized Kalman Filter Techniques," *Proceedings of the IEEE Southern Tier Technical Conference*, Inst. of Electrical and Electronics Engineers, 1990, pp. 79–87.

[6] Hashemipour, H., Roy, S., and Laub, A., "Decentralized Structures for Parallel Kalman Filtering," *IEEE Transactions on Automatic Control*, Vol. 33, No. 1, 1988, pp. 88–94. doi:10.1109/9.364

[7] Carlson, N., "Federated Square Root Filter for Decentralized Parallel Processors," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 26, No. 3, 1990, pp. 517–525. doi:10.1109/7.106130

[8] Steffes, S. R., et al., "SINPLEX: A Small Integrated Navigation System for Planetary Exploration," AAS Paper 2013-043, Breckenridge, CO, 2013.

[9] Hartkopf, S., "Entwurf und Implementierung Eines Hybriden Navigationssystems für Experimentalanwendungen," Master's Thesis, Technische Universität Darmstadt, 2013.

[10] Steffes, S. R., "Real-Time Navigation Algorithm for the SHEFEX2 Hybrid Navigation System Experiment," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2012-4990, 2012.

[11] Steffes, S. R., "Flight Results from the SHEFEX2 Hybrid Navigation System Experiment," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2012-4991, 2012.

[12] Gelb, A., *Applied Optimal Estimation*, MIT Press, Cambridge, MA, 1974, Chap. 4.

[13] John, L., Crassidis, F. L. M., and Cheng, Y., "Survey of Nonlinear Attitude Estimation Methods," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 1, 2007, pp. 12–28. doi:10.2514/1.22452