Autonomous and Resilient Management of All-Source Sensors

Juan D. Jurado and John F. Raquet
Air Force Institute of Technology

BIOGRAPHIES

Juan Jurado

Major Juan Jurado is a doctoral student at the Air Force Institute of Technology. He holds a B.S.EE from Texas A&M University, a M.S.EE from the Air Force Institute of Technology, and an M.S.FTE from the Air Force Test Pilot School. Previously, he served as Director of Engineering for the 413th Flight Test Squadron and oversaw various C-130, V-22, and H-1 flight test programs. His research interests include aircraft performance modeling, online sensor calibration, image processing, visual-inertial navigation, and statistical sensor management for multi-sensor navigation problems.

John Raquet

John Raquet is the Director of the Autonomy and Navigation Technology (ANT) Center at the Air Force Institute of Technology, where he is also a Professor or Electrical Engineering. He has been involved in navigation research for over 25 years.

ABSTRACT

Multi-sensor navigation is quickly becoming operational as more alternative sensors continue to mature. Unfortunately, increasing the number of sensors also increases the possibility of corrupting the navigation solution with incorrect measurement models and undetected sensor failures. The multi-sensor resiliency challenge remains largely unsolved, with only some sensor combinations (e.g., visual-inertial and GPS-inertial) having dedicated resiliency research. This research aims to provide multi-sensor resiliency through an autonomous sensor management framework. The proposed framework places each sensor into one of four modes: monitoring, validation, calibration, and remodeling. Each mode contains particular tasks that affect how the filter processes sensor measurements. The framework is developed by generalizing and interfacing functions found in existing research (fault detection, parameter estimation, and model selection) to achieve similar goals, but in a robust and sensor-agnostic manner. The proposed framework is compared against conventional filtering using two simulated scenarios including multiple sequential sensor failures and incorrectly modeled sensors.

INTRODUCTION

Over the past two decades, a significant portion of navigation research has been devoted to alternative means of precision navigation and timing, through the modeling and testing of non-traditional sensors (e.g., vision [1], radio [2], magnetic [3], etc.). As research matures in each of these sensor areas, multi-sensor alternative navigation is quickly becoming an operational possibility. However, with each additional sensor allowed into a navigation system comes the increased possibility of corrupting the navigation solution due to sensor model misspecification and undetected sensor failures or anomalies. Therefore, a robust method of managing sensors with questionable models is now necessary to ensure navigation solutions are accurate and resilient against sensor anomalies, sensor failures, and sensor model mismatches, all in a plug-and-play or online fashion.

Some research has been conducted in the areas of managing navigation sensors for computational requirements [4], managing self-correcting Simultaneous Localization and Mapping (SLAM) sensors [5], and validating ad-hoc sensor networks [6]. However, there are currently no frameworks, to the authors' knowledge, that formalize the definitions and interface between the processes of detecting faulty sensors, estimating sensor model parameters, and adapting sensor model functions, all in an online fashion and while preventing the navigation solution from becoming corrupted. There is, however, compartmentalized research in related areas, where common navigation challenges related to sensor modeling have been solved, albeit in limited scope and usually tuned for specific sensors. These related research areas are summarized below.

The first and arguably most critical step in resilient sensor management is fault detection. Sensor fault detection and exclusion is often accomplished in navigation through statistical analysis of measurements and measurement residuals [7][8][9][10][11][12][13][14][15][16][17]. However, the majority of these methods commonly focus on navigation problems with a single sensor or a deeply coupled sensor pair, where the measurement residuals affected by a fault belong to a known sensor, which is not a valid assumption in a multi-sensor problem. Nonetheless, detection and identification of a faulty sensor is only the first step of the problem in creating a resilient sensor management system. Since many of the detected faults may be created by model misspecification or temporary anomalies, there also exists a need to overcome the faults (and continue using the sensor in question) through the online modification of the specified sensor model.

One common method for overcoming sensor misspecifications is to estimate variable sensor model parameters that may have changed during navigation (e.g., lever arms, rotation matrices, scale factors, etc.). This type of sensor model modification is often referred to as calibration. There are a large number of online calibration examples across many sensor fields such as magnetometers [18], accelerometers [19], gyroscopes [20], lasers [21], audio sensors [22], Pitot-static sensors [23], to name a few. One such research area with significant recent advances is online calibration of a Visual-Inertial System (VINS) [24][25][26][27][28]. As mature as these calibration research areas may be, they still only tend to focus on a particular sensor or sensor combination (e.g., visual and inertial or magnetic and inertial), and often do not address the tasks of detecting the need for calibration and independently evaluating the effectiveness of the calibration results. Additionally, they do not adequately address other types of sensor model modifications that may be needed for resiliency.

A second class of methods for overcoming sensor misspecifications is to alter the functional form of the sensor model to account for missing parameters or changing environmental conditions (e.g., time-changing biases, stochastic clock errors, temperature effects, etc.). This type of sensor model modification is often referred to as adaptive estimation. Current adaptive estimation literature tends to be divided between continuous estimation of sensor and process noise covariances [29][30][31] and multiple model estimation using a finite set of competing models [32][33][34]. These techniques provide mature methods for adaptive estimation. However, they tend to be tailored for specific failure modes and require the adaptation process to run continuously. Additionally, they do not address the calibration task, and do not tend to provide an independent means for validating the adaptive estimation results.

This research proposes a novel framework that contributes both a common language and a set of critical functions and their interactions, that together provide sensor-agnostic, statistically rigorous, and resilient sensor management. The proposed framework combines the detection, identification, calibration, model selection, and independent validation functions into four interconnected modes of operation: monitoring, validation, calibration, and remodeling. In doing so, it is able to provide a greater number of resiliency functions than any of the individual research subsets alone, thereby directly enabling continued navigation operations across a greater range of sensor and/or sensor model anomalies. The complete set of resiliency functions directly enabled through the proposed framework is summarized in Table 1.

Table 1: List of resilient sensor management functions provided by the proposed framework.

Mode	Online Sensor Management Functions
Monitoring	Detect inconsistent measurement statistics Identify the affected or faulty sensor Decide when to modify existing sensor model
Validation	Validate a questionable sensor model against known sensors Verify results from parameter estimation and model selection Recover from temporary sensor failures
Calibration	Estimate selected sensor model parameters Dynamically augment and initialize necessary filter states Follow a prescribed sequence for parameter estimation
Remodeling	Select best sensor model from a list of candidates Dynamically initialize multiple-model filter bank

RELATED WORK

Though no framework combining the aforementioned functions from Table 1 been found in literature, several works have been found whose contributions were leveraged as part of the framework development process. These works are briefly described below.

Statistical hypothesis testing forms an integral part of the fault detection, model validation, and adaptive estimation tasks. In general, any hypothesis test can be stated using a Likelihood Ratio Test (LRT) [35], by assigning competing statistical distributions to each of the hypotheses in the test. Such LRTs are useful since their distribution tends to be Chi-squared [36][37], regardless of the distributions of the competing hypotheses, and especially if the competing hypotheses are assumed to be normally distributed. Integrity monitoring and fault detection and exclusion research in the area of Global Positioning System (GPS) and Inertial Navigation System (INS) integration such as [8][9][10][11][12][15][16][17][38] use LRTs and test statistics to detect faulty GPS measurements, and more importantly, predict system performance in the presence of undetected faults. Meanwhile, the work in [13] expands such methods using multiple filters to identify specific biased satellite measurements and exclude them from affecting the solution. Although not specifically shown in this paper, the implementation of our tests in the monitoring mode use a multi-filter approach such as the one described in [13] to detect and identify, not individual faulty pseudoranges, but faulty sensors in a multi-sensor system.

As previously stated, fault detection and identification are just the first two (albeit the most important) functions in a resilient sensor management framework. The goal, especially in the area of emerging and alternative sensors, is not only to detect and identify mismodeled sensors, but also dynamically modify their stated models in order to enable their continued use while protecting the navigation solution. The research in [5] proposes a multi-phased process that accomplishes multiple tasks from our proposed framework. Namely, the VINS calibration method proposed therein continually estimates camera extrinsic parameters and statistically compares, via a LRT, their short-term and long-term estimates. If the short-term and long-term estimates are statistically different, a three-phased calibration routine is initiated. Finally, the calibration routine is terminated once the covariance of the estimated parameters exhibits desired convergence criteria. In the context of our proposed framework, the research in [5] provides a VINS-centric sensor manager that provides a portion of the tasks contained in our monitoring mode, and most of the tasks contained in our calibration mode. However, it is specifically designed to work with VINSs, and lacks both multi-sensor fault detection as well as identification and independent validation of the calibration results.

When detected sensor failures are not caused by inaccurate calibration parameters, one other cause may be that the stated sensor model is either missing parts of the measurement function, or has been equipped with the wrong measurement functional form. Since a particular functional form may be wrong in an infinite number of ways, the task of finding the most appropriate sensor model function is usually solved by fitting a finite set of models in a multiple model technique such as [33][34]. Alternatively, some may choose to estimate certain stochastic model parameters continuously such as [29][30][31]. All of these methods for overcoming incomplete or incorrect model functions provide general examples of tasks contained in our remodeling mode. The specific remodeling technique shown in our example scenarios uses a filter bank with multiple parallel models, much like the aforementioned research. However, it also uses statistical model selection criteria such as Akaike Information Criterion (AIC)[39] to select the most likely model, and independently validates the selection results using our test in the validation mode.

Finally, as foreshadowed in the previous paragraphs, existing research often lacks a means of independently (or externally) validating calibration and adaptive estimation results for a particular sensor. Additionally, most navigation sensors require the estimation of additional states contained in their measurement model (e.g., clock errors, biases, scale factors, etc.). However, estimating those additional states requires allowing the sensor in question to affect the navigation solution. Moreover, even if a separate filter is used for the sensor in question, allowing the sensor to fully affect the solution of any filter will often mask any model mismatches that are inside the uncertainty of the additional state being estimated. In order to solve both of these challenges, we employed the Kalman-Schmidt / partial update equations [40] [41] in the design of our validation mode test. The partial update equations allow the estimation of sensor-specific states while preventing the measurement update from affecting core navigation states (e.g., position, velocity, attitude).

AN AUTONOMOUS AND RESILIENT SENSOR MANAGER

We now introduce a novel, autonomous method for resilient sensor management that performs all the previously identified functions by expounding on the building blocks contained in each of the individual research areas, and coherently weaving their functionality into a sensor-agnostic framework. The proposed framework, henceforth referred to as Autonomous and Resilient Management of All-source Sensors (ARMAS), statistically evaluates sensor performance and places each sensor into one of four operating modes: monitoring, validation, calibration, and remodeling. ARMAS then provides resilient sensor management by controlling how a navigation filter responds to measurements from a particular sensor based on that sensor's mode. Table 1 (shown previously) summarizes the key functionality provided by each mode of operation.

Framework Implementation

In general, the ARMAS framework is designed around an online or plug-and-play environment, applies to all navigation sensors, does not require sensor-specific tuning, and can be adapted to any filtering technique. The examples illustrated below were developed in MATLAB® using the SCORPION estimation framework [42] for filter spawning and measurement processing. It is important to note that the proposed framework was designed around the plug-and-play concept of operation, assuming sensors are serially added onto an ongoing navigation process. The only assumption imposed onto the initial ongoing navigation process is that the navigation solution is consistent (i.e., its estimates are unbiased, and the estimated error covariance matches actual performance). This assumption does not preclude a "weak" sensor (i.e., large measurement error covariance) from being the only sensor in the system. It simply dictates that if there is only a single sensor in the system when a new sensor is added, its residual statistics are assumed to be accurate. Figure 1 illustrates a proposed state transition diagram that coherently transitions sensors through the various modes of operation. The following sections expound on each of the modes of operation and provide general guidelines for proper implementation.

To facilitate further discussion on ARMAS development, we will adapt the stochastic estimation convention from [7] for multi-sensor applications. Consider a navigation problem of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left[\mathbf{x}(t), \boldsymbol{\epsilon}(t), \mathbf{u}(t), t\right] + \mathbf{G}(t)\mathbf{w}(t), \tag{1}$$

where $\mathbf{x}$ is the $N \times 1$ navigation state vector containing the vehicle's core navigation states (position, velocity, attitude, time, etc.), $\boldsymbol{\epsilon}$ is an $M \times 1$ vector containing additional states needed to account for measurement errors, $\mathbf{u}$ is the control input vector, $\mathbf{G}$ is an $(N + M) \times W$ linear operator, and $\mathbf{w}$ is a $W \times 1$ white Gaussian noise process with a $W \times W$ continuous process noise strength matrix $\mathbf{Q}$. The discretized [43] non-linear system is then solved using the Extended Kalman Filter (EKF) algorithm [7] [44]. The state estimates are propagated using the (possibly) non-linear state dynamics model, and updated using measurements from available sensors. In a multi-sensor environment with J sensors, the jth sensor provides Z-dimensional discrete measurements, $\mathbf{z}_k^{[j]}$, at time t_k, which are modeled as

$$\mathbf{z}_k^{[j]} = \mathbf{h}^{[j]}\left[\mathbf{x}(t), \boldsymbol{\epsilon}^{[j]}(t), \mathbf{u}(t), t, \mathbf{p}^{[j]}\right] + \mathbf{v}_k^{[j]}, \tag{2}$$

where $\mathbf{h}^{[j]}$ is a (possibly) non-linear measurement function for sensor j, $\boldsymbol{\epsilon}^{[j]}$ is a $L \times 1$ ($L \leq M$) subset of $\boldsymbol{\epsilon}$ consisting of the additional states required for processing measurements for sensor j (e.g., a clock error process, constant bias, etc.), $\mathbf{p}^{[j]}$ is a $P \times 1$ vector of observable model parameters for $\mathbf{h}^{[j]}$ selected by the user (e.g. a lever arm, scale factor, etc.), and $\mathbf{v}_k^{[j]}$ is a $Z \times 1$ discrete white Gaussian noise process with covariance matrix $\mathbf{R}_k^{[j]}$. Given the estimated quantities $\hat{\mathbf{x}}_k^-$, $\hat{\boldsymbol{\epsilon}}_k^{[j]^-}$, $\hat{\mathbf{p}}^{[j]}$, the $Z \times 1$ measurement residual, $\mathbf{r}_k^{[j]}$, at time $t = t_k$ for sensor j is given by

$$\mathbf{r}_k^{[j]} = \mathbf{z}_k^{[j]} - \mathbf{h}^{[j]} \left[\hat{\mathbf{x}}_k^-, \hat{\boldsymbol{\epsilon}}_k^{[j]^-}, \mathbf{u}_k, t_k, \hat{\mathbf{p}}_k^{[j]} \right]. \tag{3}$$

Additionally, the residual vector from (3) is expected to follow the distribution

$$\mathbf{r}_k^{[j]} \sim \mathcal{N}\left(\underset{Z\times 1}{\mathbf{0}}, \mathbf{S}_k^{[j]} \right), \tag{4}$$

$$\mathbf{S}_k^{[j]} = \mathbf{H}_k^{[j]} \mathbf{P}_k^- \mathbf{H}_k^{[j]^{\mathrm{T}}} + \mathbf{R}_k^{[j]}, \tag{5}$$

where $\mathbf{H}_k^{[j]}$ is the $Z \times (N + M)$ Jacobian of $\mathbf{h}^{[j]}$ about the current estimate, and $\mathbf{P}_k^-$ is the $(N + M) \times (N + M)$ state estimation error covariance matrix at time $t = t_k$. In this context, the goal of ARMAS is to ensure incoming measurements $\mathbf{z}_k^{[j]}$ truly adhere to their stated models by analyzing the statistical distribution of their residuals; and if not, protecting the core navigation solution, $\hat{\mathbf{x}}$, while attempting to modify $\boldsymbol{\epsilon}^{[j]}$ and/or $\mathbf{p}^{[j]}$ to enable continued sensor use.

Sensor Initialization

As previously stated, ARMAS places each sensor into one of four operating modes plus a failed state. In a plug-and-play environment with an ongoing navigation process, each new sensor is initialized into one of two modes: monitoring or validation. This initial placement is based on how confident users may be in the current model available for a particular sensor. For example, sensors that have well understood models and stochastic processes, such as a GPS receiver, could be considered “trusted”, and placed directly into monitoring mode, while emerging alternative sensors with more questionable error models could be considered “untrusted” and placed into validation mode. To take advantage of this functionality, an ARMAS user can specify the initial trust for each sensor (e.g., trusted or untrusted), or provide a default setting for all new sensors.

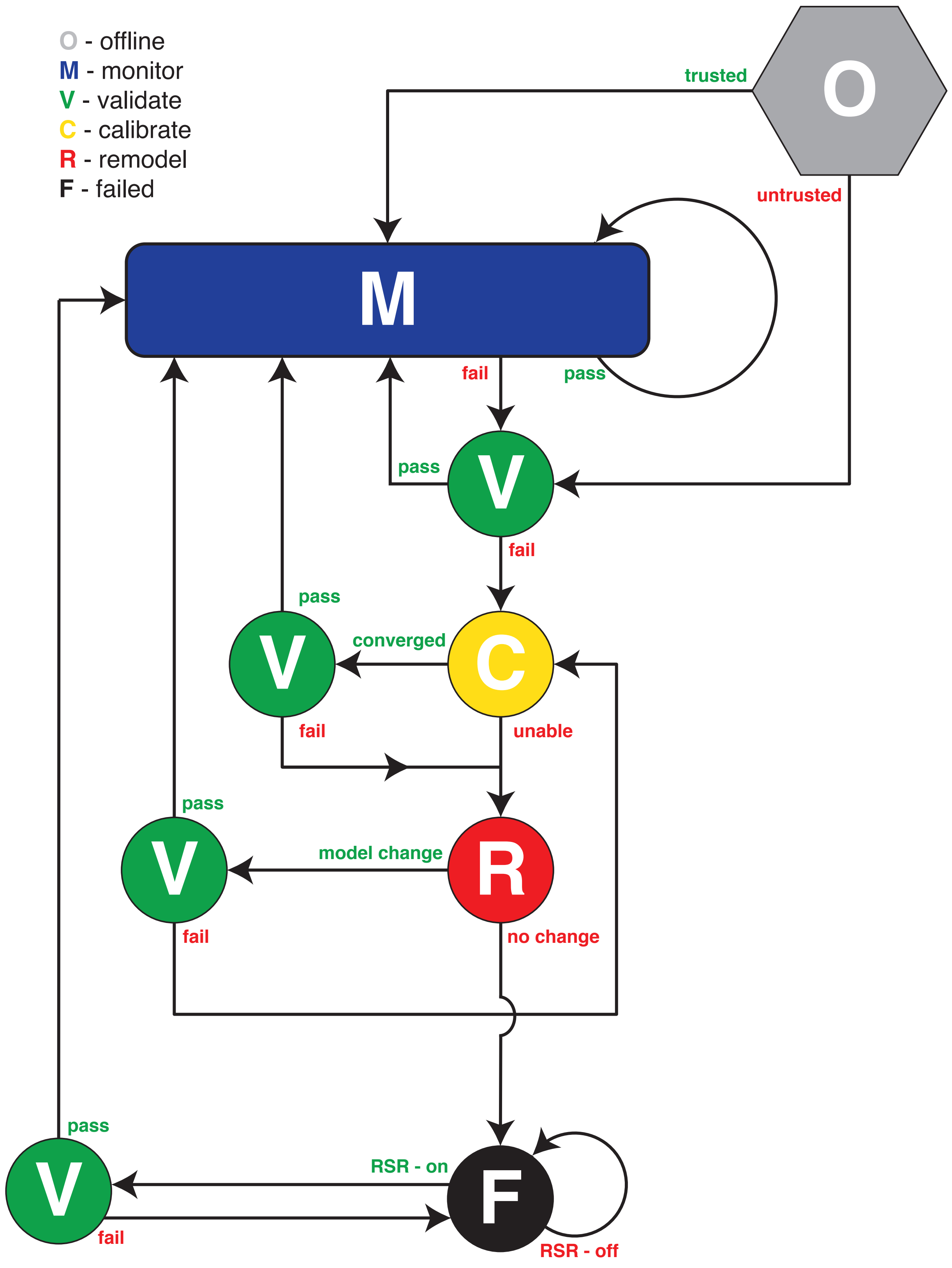

Figure 1: Proposed state transition diagram for ARMAS framework. New sensors (offline) begin in either validate or monitor mode. A failed monitoring test starts a calibration and remodeling loop, whose effectiveness is evaluated by the validation test. The loop continues until the validation test is passed or the sensor model selection is unchanged. Failed sensors can be periodically re-validated to recover from temporary anomalies using Resilient Sensor Recovery (RSR).

Monitoring Mode

Monitoring mode is used to detect and identify faulty sensors in a multi-sensor problem. In this context, a faulty sensor refers to a sensor whose stated model is not consistent with its observed performance, which could be caused by temporary or permanent failures, as well as dynamic changes to the sensing environment (e.g., atmosphere, terrain, multi-path, etc.). Sensors in monitoring mode are able to fully affect the navigation solution provided to the user since they are trusted. This poses a challenge to detecting model divergence using a single-filter solution, since the effects of a mismodeled sensor on the navigation solution are not easily attributable to a particular sensor. An example of this problem is a strong (i.e., low measurement noise strength) sensor with an un modeled bias that "pulls" the filter solution away from truth to absorb the bias. In that case, the faulty sensor's residuals would be statistically valid, while potentially making the residuals from weaker yet not faulty sensors invalid. Any method that can robustly detect and attribute sensor faults can be used in this mode. For our particular implementation, we adapted a multi-filter approach similar to that described in [13] to apply beyond satellite pseudorange measurements. In general, our current approach is two-phased, and relies on running K parallel sub-filters, each filter excluding one of the $K \leq J$ sensors found in monitoring mode. In the first phase, measurement residuals are collected between each trusted sensor and each monitoring sub-filter using (3). This operation produces K^2 residual collections which can be individually evaluated using a statistical hypothesis test such as the χ^2 test [38]. In the second phase, if any of the χ^2 tests failed, the culprit sensor is found by assuming only a single sensor has failed and deducing the failure pattern from the $K \times K$ results matrix. This process can be expanded to assume more than one failed sensor by increasing the number of sub-filters and the number of excluded sensors per filter. Residuals are collected continuously for every sensor in monitoring mode, with a test decision produced for each sensor and at a user-defined rate. The test is passed if the sensor in question is not found to be a culprit, and failed otherwise. If the test is passed, the sensor in question remains in monitoring mode. Otherwise, the sensor is no longer allowed to affect the solution and is placed into validation mode, where its stated model is independently validated against the other (trusted) sensors. To take advantage of monitoring functionality, an ARMAS user needs to specify the following:

1. The monitoring period (e.g., number of samples or time elapsed between tests).
2. The statistical significance level (i.e., probability of false alarm).

Validation Mode

Validation mode is used to statistically validate an untrusted sensor model using information from trusted sensors. Sensors in validation mode are only able to affect state estimates for their unique states, $\boldsymbol{\epsilon}^{[j]}$, while the core navigation states, $\mathbf{x}$, are only affected by trusted sensors (i.e., sensors in monitoring mode). This poses a challenge when processing measurements from an untrusted sensor since we are to consider the stochastic distribution of all filter states while only allowing an untrusted measurement update from sensor j to affect $\boldsymbol{\epsilon}^{[j]}$. An example of this challenge would be estimating a receiver clock error without allowing the receiver to affect the navigation solution, and while preventing the clock error estimate from absorbing any other missing error sources. To solve this challenge in a plug-and-play environment, we initialize a validation sub-filter using a copy of the main filter whenever a (untrusted) sensor goes into validation mode. Using the Schmidt or partial update equations [40][41] in the sub-filter, measurement updates from an untrusted sensor j are only able to affect $\boldsymbol{\epsilon}^{[j]}$, while measurement updates from sensors in monitoring mode (i.e., trusted) are able to affect all filter states. Residuals are then collected between the untrusted sensor and the validation sub-filter using (3), and tested for likelihood of occurrence using any statistical hypothesis test. For our particular implementation, we developed a general LRT using residual Malanahobis distances [36]. Similar to the behavior in monitoring mode, residuals are collected continuously for every sensor in validation mode, with a test decision produced at a user-defined rate. The mode transition from validation mode is determined by both the results of the validation LRT and the previous ARMAS mode of the sensor in question. As illustrated in Figure 1, validation mode is used to externally validate sensor models after failing a monitoring test, completing a calibration sequence, and completing a remodeling selection. Additionally, users may opt to periodically place permanently failed sensors back into validation mode to check for passage of temporary anomalies that might have caused an insuperable failure, which we refer to as Resilient Sensor Recovery (RSR). To take advantage of validation functionality, an ARMAS user needs to specify the following:

1. The validation period (e.g., number of samples or time elapsed between tests).
2. The statistical significance level (i.e., probability of false alarm).

3. Which states used in $\mathbf{h}^{[j]}$ are core navigation states (i.e., $\mathbf{x}$).
4. Which states used in $\mathbf{h}^{[j]}$ are sensor-unique states (i.e., $\boldsymbol{\epsilon}^{[j]}$).
5. If RSR is enabled, the RSR period (i.e., how often to attempt post-fail validation).

Calibration Mode

Calibration mode is used to dynamically re-estimate variable model parameters when a sensor model has been identified as faulty. Sensors enter calibration mode after having failed an initial or post-monitoring validation test, and are considered untrusted until they can be validated and placed back into monitoring mode. In calibration mode, the functional form of the sensor model is assumed correct, but certain model parameters, $\mathbf{p}^{[j]}$, inside the measurement model function from (2) are to be re-estimated. The parameter estimation may also require specific sequencing (i.e., estimate subsets of $\mathbf{p}^{[j]}$ in a specified order) in order to maintain observability, which in turn requires specifying sequencing criteria, or a method for determining when a step in the sequence is complete. An example of this type of problem would be the camera calibration algorithm described in [25], where the camera extrinsic parameters (lever arm and Euler angles) are estimated separately (Euler angles first, then lever arm) in order to maintain observability, and the transition between sequence steps is driven by convergence of the associated state covariance matrix. For our particular implementation, we generalized the calibration process into a set of simple instructions provided by users such that any sensor calibration method that can be expressed in terms of a sequenced state vector augmentation and corresponding sequence transition criteria can be easily and autonomously executed. The parameter estimation is performed using a calibration sub-filter that is initialized based on a copy of the main filter whenever a sensor enters calibration mode. Once all steps in the stated calibration sequence are completed, the sensor in question is placed back into validation mode to externally validate the calibration results against trusted sensors as described above. It is also important to note here that since sensors in calibration mode are considered untrusted, their measurement updates during a calibration sequence are only able to affect estimates for $\boldsymbol{\epsilon}^{[j]}$ and $\mathbf{p}^{[j]}$, which tends to increase observability on $\mathbf{p}^{[j]}$ since estimates for $\mathbf{x}$ are provided by trusted sensors. For this reason, a calibration performed using ARMAS can be considered an external or independent calibration. To take advantage of calibration functionality, an ARMAS user needs to specify the following:

1. Which parameters in $\mathbf{h}^{[j]}$ are to be estimated during calibration (i.e., $\mathbf{p}^{[j]}$).
2. The initial estimate for each parameter (i.e., $\mathbf{p}_0^{[j]}(i),\ i = 1, \ldots, P$).
3. The initial uncertainty for each parameter (i.e., $\boldsymbol{\sigma}_0^{[j]}(i),\ i = 1, \ldots, P$).
4. The sequence group for each parameter (i.e., when to start estimating).
5. Transition criteria for each parameter (i.e., when to stop estimating).

Subsequently, if the sensor in question enters calibration mode, ARMAS automatically augments the filter states using the provided initial estimates and uncertainties for the first group in the sequence, then transitions to the next group once every group member has met its transition criteria, until all groups are completed.

Remodeling Mode

Remodeling mode is used to dynamically modify the functional form of the measurement model when a sensor model has been identified as faulty. Sensors enter remodeling mode after failing a post-calibration validation test or if a calibration routine was not provided. Similar to other modes, sensors in remodeling mode are considered untrusted until they can be validated and placed back into monitoring mode. In remodeling mode, the functional form of the sensor model is now assumed incorrect. This poses a challenge since a measurement function $\mathbf{h}^{[j]}$ could be incorrectly stated in an infinite number of ways. To solve this, we leverage user experience for each sensor application to create a reasonable and finite set of S model options that are dynamically evaluated if the sensor in question enters remodeling mode. Then, we use a statistical test to select the best model from the set. An example of this type of problem would be a VINS user that initially only models radial lens distortion, but includes the option to also model tangential lens distortion if the camera sensor ever enters the remodeling mode. Similarly, a laser range finder user may elect to initially attempt a First Order Gauss-Markov (FOGM) error model, but include the option to try second-order, constant bias, or the combination thereof, if the sensor ever enters remodeling mode. Any method that can evaluate multiple models and statistically select the most appropriate one can be used in this mode. For our particular implementation, we leveraged Multiple Model Adaptive Estimation (MMAE) research such as [32][33][34], but used Akaike Information Criterion (AIC) [39] for the model selection decision since it balances model complexity with error reduction. Additionally, we only evaluate multiple models when the sensor in question enters remodeling mode, spawning S remodeling sub-filters initialized as copies of the main filter, and collecting residuals using (3) until all remodeling termination criteria are met. Once a model selection decision is made, the sensor in question along with the selected model are placed into validation mode. Similarly to calibration mode, sensors in remodeling mode are untrusted and only able to affect estimates of $\boldsymbol{\epsilon}^{[j]}$ and $\mathbf{p}^{[j]}$, which strengthens the model selection process since it is heavily influenced by external and trusted sensors. To take advantage of remodeling functionality, an ARMAS user needs to specify the following:

1. A list of candidate measurement models (i.e., $\mathbf{h}_i^{[j]}\ i = 1, \ldots, S$).

2. The initial estimate for all sensor-unique states in each model (i.e., $\boldsymbol{\epsilon}_{i_0}^{[j]}\ i = 1, \ldots, S$).

3. The initial covariance for all sensor-unique states in each model (i.e., $\boldsymbol{\Sigma}_{i_0}^{[j]},\ i = 1, \ldots, S$).

4. The remodeling termination criteria (e.g., number of samples, time period, covariance, etc.).

Subsequently, if the sensor in question enters remodeling mode, ARMAS automatically spawns S parallel filters using the main filter statistics for the core states, $\mathbf{x}$, and the provided initial estimates and covariances for each set of sensor-specific states in each candidate model.

Implementation Summary

As shown in the previous sections, ARMAS provides an autonomous method for implementing various sensor model management functions given a minimal set of additional specifications for each sensor beyond the usual modeling requirements. It is important to note, certain functions including multi-sensor fault detection and identification as well as RSR, are available without any additional sensor-centric specifications, and using only default settings. Table 2 summarizes the additional information required to enable ARMAS functionality for each mode of operation.

Example Scenarios

This section describes two example scenarios that relied on the ARMAS framework for resilient sensor management. It is important to note that these are simply two examples of how the proposed framework could be used. A user of the framework has the flexibility to adopt the framework to the specific problem at hand. In both scenarios, the use of ARMAS enabled continued operations in cases that would have resulted in either significant navigation solution errors or irrecoverable sensor failures. Consider a two-dimensional navigation problem with two vehicles (Aircraft 1 and Aircraft 2) obtaining their navigation solutions from an EKF that uses a kinematic model given by

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{\mathbf{x}}_p(t) \\ \dot{\mathbf{x}}_v(t) \\ \dot{\mathbf{x}}_a(t) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_v(t) \\ \mathbf{x}_a(t) \\ -\frac{1}{\tau_a}\mathbf{x}_a(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{w}(t) \end{bmatrix}, \tag{6}$$

where $\mathbf{x}_p$ is the vehicle's two-dimensional position in [m], $\mathbf{x}_v$ is the two-dimensional velocity in [m/s], $\mathbf{x}_a$ is the two-dimensional acceleration in [m/s^2], which is driven by a FOGM process with time constant $\tau_a = 90$ [s], and $\mathbf{w}$ is a two-dimensional white Gaussian noise process with $E\left[\mathbf{w}\mathbf{w}^{\mathrm{T}}\right] = (1.5 \times 10^{-3})^2 \underset{2\times 2}{\mathbf{I}}$ [m^2/s^6]. The initial state estimate, $\hat{\mathbf{x}}(0)$, and state error covariance, $\mathbf{P}(0)$, for both vehicles at the beginning of each scenario is given by

$$\hat{\mathbf{x}}(0) = \begin{bmatrix} 0 & 0 & 100 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}, \tag{7}$$

$$\mathbf{P}(0) = \mathrm{diag}\Big(\begin{bmatrix} 1 & 1 & 10 & 10 & 1.5 \times 10^{-3} & 1.5 \times 10^{-3} \end{bmatrix}^2\Big). \tag{8}$$

Table 2: ARMAS implementation summary.

Parameter name	**Symbol**	**Required by**[1]				**Example specification**
		M	**V**	**C**	**R**	
Significance level	α	X	X			0.05
Monitoring period		X				20 [s]
Validation period			X			60 [s]
RSR period			X			30 [s]
Core navigation states	$\mathbf{x}$		X	X	X	EKF States: 1, 2, 3, 4, 5, 6
Sensor-unique states	$\boldsymbol{\epsilon}^{[j]}$		X	X	X	EKF States: 7, 8
Calibration parameters[2]	$\mathbf{p}^{[j]}$			X		$\mathbf{h} = \left[\begin{array}{cc} \mathbf{p}(1)\mathbf{x}(1) & \mathbf{p}(2)\mathbf{x}(2) \end{array} \right]^{\mathrm{T}}$
Initial parameter values[2]	$\mathbf{p}_0^{[j]}$			X		$\mathbf{p}_0 = \left[\begin{array}{cc} 1 & 1 \end{array} \right]^{\mathrm{T}}$
Initial parameter uncertainty[2]	$\boldsymbol{\sigma}_0^{[j]}$			X		$\boldsymbol{\sigma}_0 = \left[\begin{array}{cc} 10 & 10 \end{array} \right]^{\mathrm{T}}$
Calibration sequence[2]				X		Group 1: $\{\mathbf{p}(1)\}$
						Group 2: $\{\mathbf{p}(2)\}$
Transition criteria[2]				X		$\mathbf{p}(1)$: 300 [s]
						$\mathbf{p}(2)$: 300 [s]
Candidate models[3]	$\mathbf{h}_i^{[j]}$				X	$\mathbf{h}_1 = [\mathbf{x}(1) + b_1\ , \mathbf{x}(2)]^{\mathrm{T}}$
						$\mathbf{h}_2 = [\mathbf{x}(1) + b_1\ , \mathbf{x}(2) + b_2]^{\mathrm{T}}$
Initial state estimates[3]	$\boldsymbol{\epsilon}_{i_0}^{[j]}$				X	$\boldsymbol{\epsilon}_{1_0} = 0$
						$\boldsymbol{\epsilon}_{2_0} = \left[\begin{array}{cc} 0 & 0 \end{array} \right]^{\mathrm{T}}$
Initial state covariances[3]	$\boldsymbol{\Sigma}_{i_0}^{[j]}$				X	$\boldsymbol{\Sigma}_{1_0} = 100$
						$\boldsymbol{\Sigma}_{2_0} = \left[\begin{array}{cc} 100 & 0 \\ 0 & 100 \end{array} \right]$
Termination criteria[3]					X	100 residual samples

[1] M - Monitoring, V - Validation, C - Calibration, R - Remodeling

[2] Example: 2D scale factor, each dimension estimated separately, time-based sequencing between dimensions

[3] Example: Model 1 adds 1D bias, Model 2 adds 2D bias, both models use sample size for termination

Each vehicle obtains discrete measurement updates from three sensors (Sensor A, Sensor B, and Sensor C). Sensor A is a two-dimensional position sensor with a model given by

$$\mathbf{z}_k^{[A]} = \mathbf{s} \circ \mathbf{x}_{p_k} + \mathbf{v}_k^{[A]}, \tag{9}$$

$$\mathbf{v}_k^{[A]} \sim \mathcal{N}\left(\underset{2\times1}{\mathbf{0}}, 100^2 \underset{2\times2}{\mathbf{I}} \right), \tag{10}$$

where $\mathbf{s}$ is a two-dimensional scale factor, $\circ$ denotes the Hadamard product, and $\mathbf{I}$ is an identity matrix. Sensor B is a two-dimensional, eight-satellite pseudorange sensor with a model given by

$$\mathbf{z}_k^{[B]} = \left[\begin{array}{c} \|\mathbf{t}_1 - \mathbf{x}_{p_k}\| \\ \vdots \\ \|\mathbf{t}_8 - \mathbf{x}_{p_k}\| \end{array} \right] + \left[\begin{array}{c} b_k \\ \vdots \\ b_k \end{array} \right] + \mathbf{v}_k^{[B]}, \tag{11}$$

$$\mathbf{v}_k^{[B]} \sim \mathcal{N}\left(\underset{8\times1}{\mathbf{0}}, 20^2 \underset{8\times8}{\mathbf{I}} \right), \tag{12}$$

where $\mathbf{t}_i$ is the two-dimensional position of satellite i, $\mathbf{x}_{p_k}$ is the two-dimensional position of the vehicle at time t_k, and b_k is a FOGM process simulating a receiver clock error with time constant $\tau_B = 3600$ [s] and $\sigma^2 = 8000^2$ [m^2]. Finally, Sensor C is a two-dimensional velocity sensor with the model

$$\mathbf{z}_k^{[C]} = \mathbf{x}_{v_k} + \mathbf{v}_k^{[C]}, \tag{13}$$

$$\mathbf{v}_k^{[C]} \sim \mathcal{N}\left(\underset{2\times1}{\mathbf{0}}, 50^2 \underset{2\times2}{\mathbf{I}} \right). \tag{14}$$

Table 3: Key events and RSS position error comparison, Example 1.

Sensor	Event	Time [min]	RSS Position Error		
			Aircraft 1 [m]	Aircraft 2 [m]	% Change
All	Start	0	0	0	0
B	Anomaly: On	5	2.2	2.2	0
B	Anomaly: Off	15	369.6	10.9	-97.1
All	End	20	4.8	4.8	0

Example 1: Temporary sensor anomaly

In this scenario, ARMAS was used to detect the presence of a temporary anomaly in Sensor B. Aircraft 1 was equipped with a standard EKF while Aircraft 2 used ARMAS. ARMAS specifications for this scenario were as follows:

- The significance level for monitoring and validation was set to 0.05.
- The monitoring period was defined by time elapsed and set to 20 [s].
- The validation period was defined by time elapsed and set to 60 [s].
- The RSR period was defined by time elapsed and set to 60 [s].
- The core navigation states were defined as all states shown in (6) (e.g., $\mathbf{x}_p, \mathbf{x}_v, \mathbf{x}_a$).
- Sensor B included a sensor-unique state, b_k, as defined in (11).
- There were no calibration routines or remodeling options specified for any sensor.

At the start of the scenario, all sensors in Aircraft 2 were in monitoring mode. After five minutes, a temporary anomaly defined by an unmodeled bias, which grew from 0 [m] to 1500 [m] over 10 minutes, was applied to the fourth entry in $\mathbf{z}_k^{[B]}$ from (11). As shown in Figure 2, Aircraft 1 had no means to detect the growing Sensor B bias, causing its solution to drift from the truth. In contrast, Aircraft 2 was equipped with ARMAS and therefore, was able to quickly identify the mismatch between the solution versions across the three sensors, and identify Sensor B as the cause of the divergence. Next, as shown in Figure 3, Sensor B failed the monitoring test, and was placed into validation mode, where it also failed the validation test. Since no calibration routine or remodeling options were provided, Sensor B transitioned from validation mode to a failed state, where RSR periodically validated its performance. Aircraft 2 continued navigation using only Sensor A and Sensor C, as indicated by the increase in position covariance in Figure 2. After some time, the two aircraft physically transitioned away from the anomaly area, and the solution in Aircraft 1 quickly converged towards the truth. Meanwhile, Aircraft 2 continued to navigate using only Sensor A and Sensor C until RSR led to a passing validation test. Once Sensor B was validated, Aircraft 2 returned to navigation with all sensors in monitoring mode. Table 3 compares the solution performance between the two aircraft for this scenario. Note the large difference in Root Sum Squared (RSS) position error between the two aircraft at the point the temporary anomaly ended, as shown in Table 3.

Example 2: Multiple sequential faults

In this scenario, ARMAS was used to validate and remodel an untrusted sensor model for Sensor B, as well as detect the need to calibrate Sensor A. Again, Aircraft 1 was equipped with a standard EKF while Aircraft 2 used ARMAS. ARMAS specifications for this scenario were as follows:

- The significance level for monitoring and validation was set to 0.05.
- The monitoring period was defined by time elapsed and set to 20 [s].
- The validation period was defined by time elapsed and set to 60 [s].
- The RSR period was defined by time elapsed and set to 60 [s].

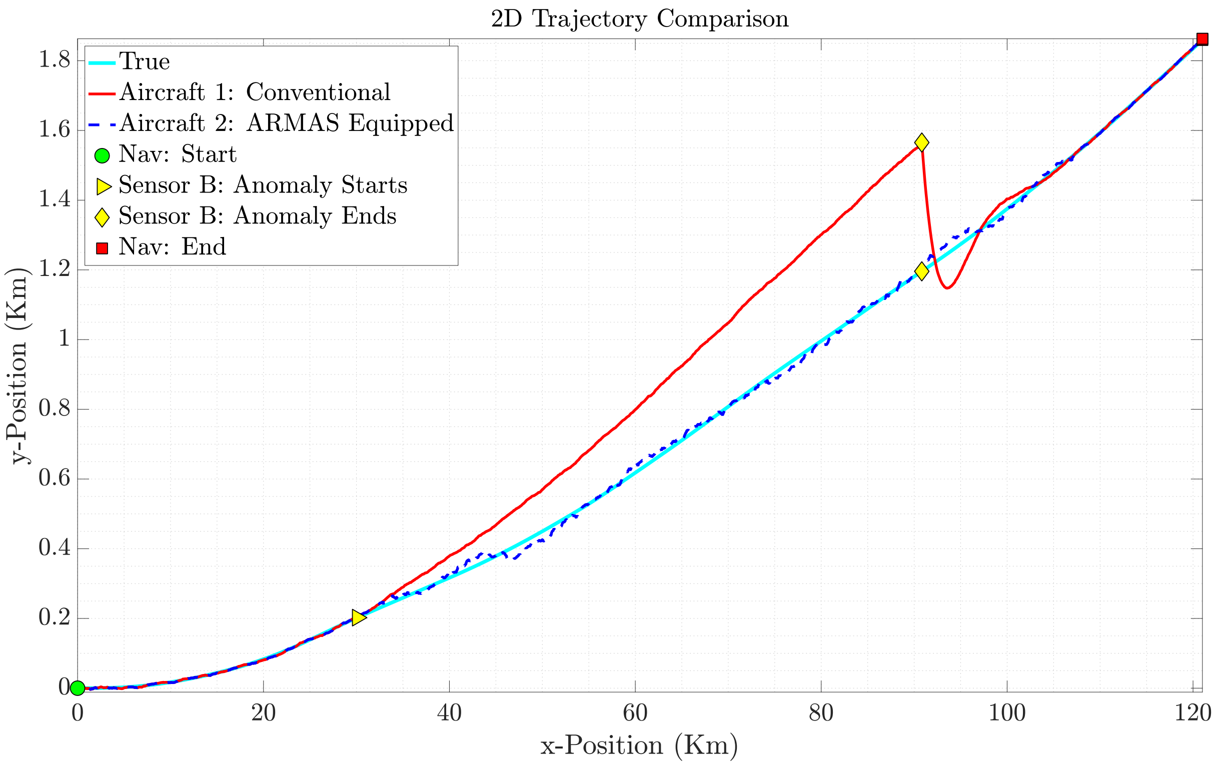

Figure 2: Trajectory comparison between Aircraft 1 and Aircraft 2, Example 1.

- The core navigation states were defined as all states shown in (6) (e.g., $\mathbf{x}_p, \mathbf{x}_v, \mathbf{x}_a$).
- Sensor B included a sensor-unique state, b_k, as defined in (11).
- Sensor A was equipped with a calibration routine defined by

$$\mathbf{p}^{[A]} = \begin{bmatrix} \mathbf{s}(1) & \mathbf{s}(2) \end{bmatrix}^{\mathrm{T}}, \tag{15}$$

$$\mathbf{h}^{[A]} = \begin{bmatrix} \mathbf{s}(1)\mathbf{x}_p(1) & \mathbf{s}(2)\mathbf{x}_p(2) \end{bmatrix}^{\mathrm{T}}, \tag{16}$$

$$\mathbf{p}_0^{[A]} = \begin{bmatrix} 1 & 1 \end{bmatrix}^{\mathrm{T}}, \tag{17}$$

$$\boldsymbol{\sigma}_0^{[A]} = \begin{bmatrix} 10 & 10 \end{bmatrix}^{\mathrm{T}}, \tag{18}$$

where the calibration sequence requires estimation of $\mathbf{p}(1)$ for 150 [s] first, then $\mathbf{p}(2)$ for an additional 150 [s].

- Sensor B was equipped with $S = 8$ remodeling candidates defined by

$$\mathbf{h}_1^{[B]} = \mathbf{h}^{[B]} + \begin{bmatrix} c & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{19}$$

$$\mathbf{h}_2^{[B]} = \mathbf{h}^{[B]} + \begin{bmatrix} 0 & c & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{20}$$

$$\vdots \tag{21}$$

$$\mathbf{h}_8^{[B]} = \mathbf{h}^{[B]} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & c \end{bmatrix}, \tag{22}$$

where $\mathbf{h}^{[B]}$ is the baseline measurement model defined in (11), and the constant bias, c, has initial statistics given by

$$c_0 \sim \mathcal{N}\left(0 \text{ [m]}, 1000^2 \text{ [m}^2\text{]}\right). \tag{23}$$

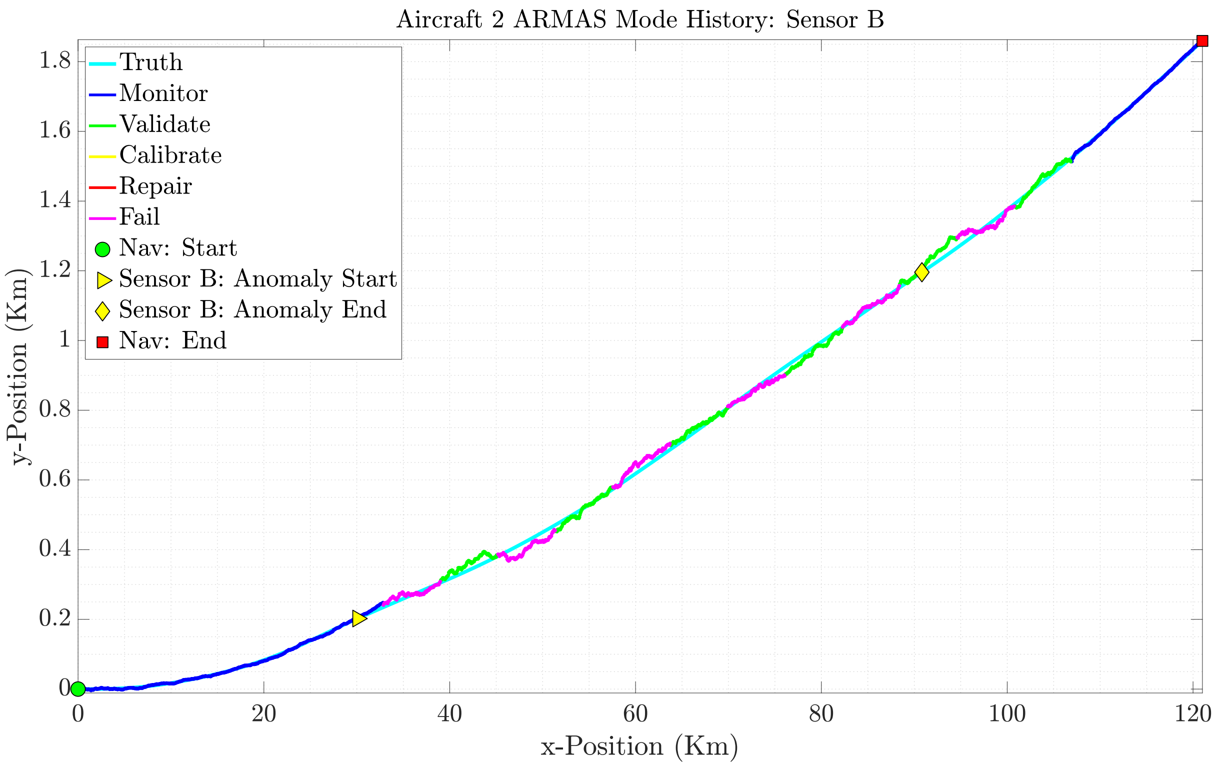

Figure 3: ARMAS mode history for Sensor B, Example 1.

At the start of the scenario, only Sensor A and Sensor C were online in both vehicles. Additionally, both online sensors in Aircraft 2 were in monitoring mode. As shown in Figure 4, Sensor B was initialized after five minutes. However, the sensor developers were not confident in the sensor model from (11), and provided ARMAS an additional eight model versions, each modeling the addition of a constant bias to a particular satellite. Meanwhile, Aircraft 1 was limited to using (11) as given. The actual measurements from Sensor B included a 1500 [m] constant bias added to the fourth entry in $\mathbf{z}_k^{[B]}$ from (11). As shown in Figure 5, the sensor model provided for Sensor B was incomplete, causing the solution in Aircraft 1 to shift away from the truth. Meanwhile, Aircraft 2 used the validation mode in ARMAS to recognize the model mismatch without compromising its navigation solution. Since there were no calibration routines provided for Sensor B, it transitioned into remodeling mode, where all model options were evaluated in parallel, while continuing to navigate using the other two sensors. The model selection from the remodeling mode was then successfully validated, placing Sensor B into monitoring mode. Five minutes later, an aircraft maneuver changed the variable scale factor on Sensor A from $\mathbf{s} = [1 \quad 1]^{\mathrm{T}}$ to $\mathbf{s} = [1.2 \quad 1.3]^{\mathrm{T}}$. As shown in Figure 4, the change did not affect Aircraft 1 since the effect on the navigation solution from Sensor A was attenuated by the measurement updates from the other two sensors. Conversely, ARMAS in Aircraft 2 detected the growing residuals and identified Sensor A as the source. As shown in Figure 6, Sensor A transitioned from monitoring to calibration mode, where the provided scale factor calibration sequence was augmented into the navigation state. The newly calibrated Sensor A then passed the validation test and was placed back into monitoring mode. Finally, ten minutes later, Sensor B was taken offline. At that point, with only Sensor A and Sensor C available to provide additional information, the solution in Aircraft 1 began to exhibit the effects of the un-calibrated Sensor A. Meanwhile, Aircraft 2 continued to operate nominally since it had previously detected the need for and successfully executed the calibration of Sensor A. Table 4 compares the solution performance between the two aircraft for this scenario. Note the diverging performance in terms of RSS position error between the two aircraft, especially after the un-calibrated Sensor A becomes the only source of position information in Aircraft 1.

Table 4: Key events and RSS position error comparison, Example 2.

Sensor	**Event**	**Time [min]**	**RSS Position Error**		
			Aircraft 1 [m]	**Aircraft 2 [m]**	**% Change**
All	Start	0	0	0	0
B	Online	5	15.9	15.9	0
A	Scale factor	10	370.3	5.8	-98.4
B	Offline	20	438.9	2.5	-99.4
All	End	25	2.78×10^4	22.8	-99.9

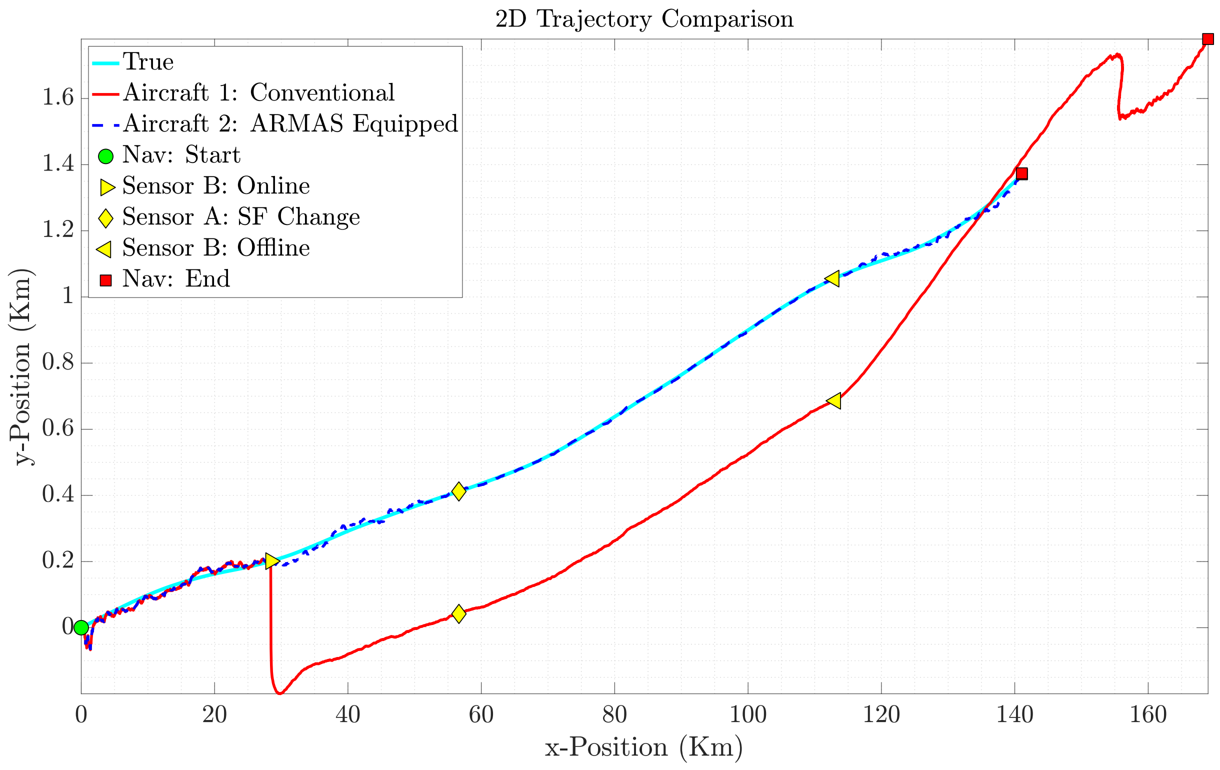

Figure 4: Trajectory comparison between Aircraft 1 and Aircraft 2, Example 2.

CONCLUSIONS

This research has introduced a novel sensor management framework that provides sensor-agnostic autonomous and resilient sensor management for alternative multi-sensor navigation problems. The proposed framework, named Autonomous and Resilient Management of All-source Sensors (ARMAS), provides a breadth of sensor management functions across four modes of operation: monitoring, calibration, remodeling, and validation. Using a coherent interconnection between these modes, ARMAS was shown to provide resilient and autonomous sensor management across two example multi-sensor navigation scenarios that required a combination of fault detection and identification, online parameter calibration, multiple-model selection, and sensor model validation. In the two examples provided, a vehicle equipped with the ARMAS framework exhibited up to 99.9% less position RSS error during temporary sensor anomalies and multiple sequential sensor failures, when compared to a non-ARMAS equipped vehicle. Future work in this area includes continued development of the novel methods for multi-sensor fault detection and sensor model validation used in the monitoring and validation modes as well as multi-trial Monte Carlo performance analysis using actual and simulated multi-sensor navigation data.

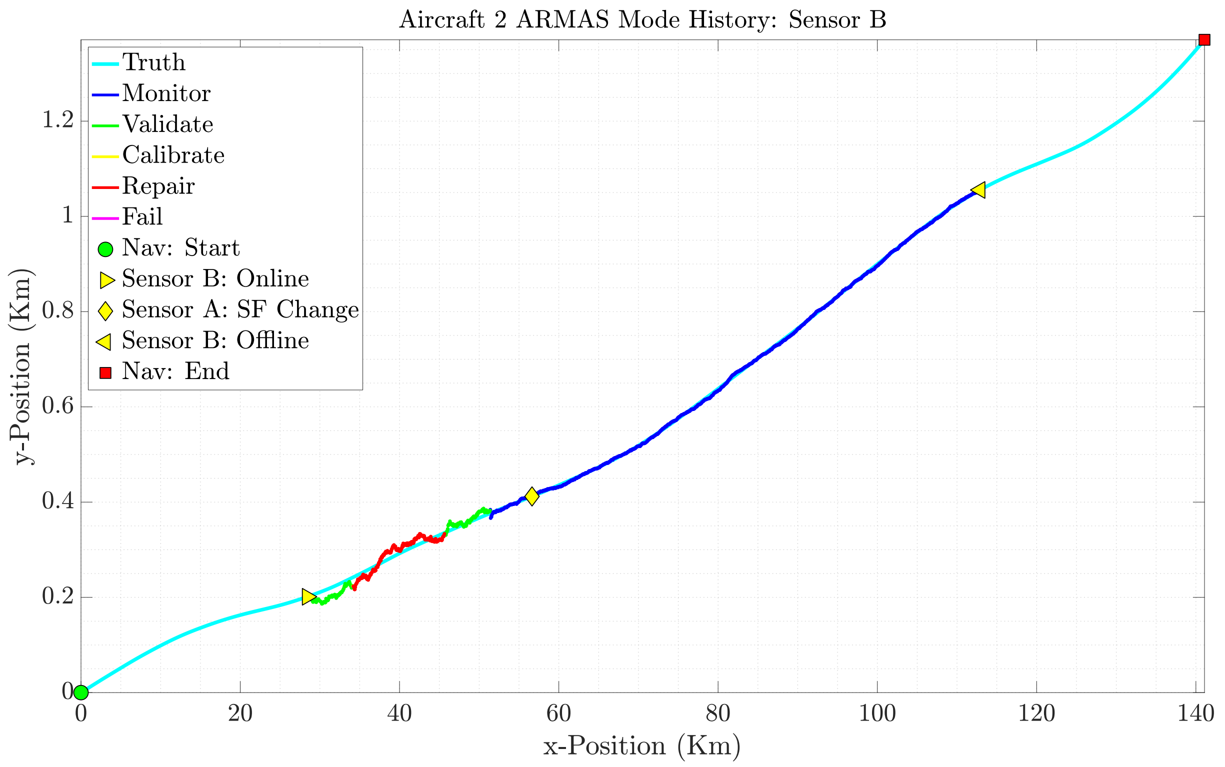

Figure 5: ARMAS mode history for Sensor B, Example 2.

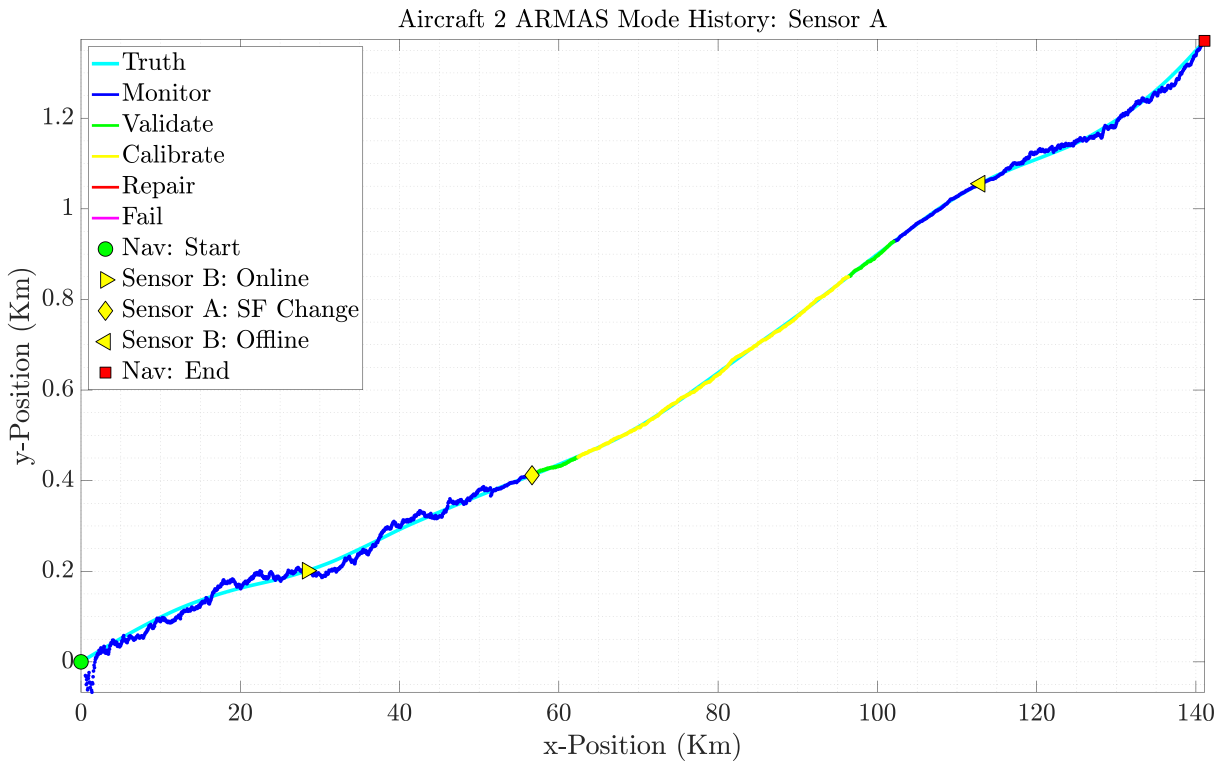

Figure 6: ARMAS mode history for Sensor A, Example 2.

DISCLAIMER

REFERENCES

[1] D. T. Venable, "Improving real world performance of vision aided navigation in a flight environment," tech. rep., Air Force Institute of Technology WPAFB, 2016.

[2] J. Curro and J. Raquet, "Navigation using vlf environmental features," in *Position, Location and Navigation Symposium (PLANS), 2016 IEEE/ION*, pp. 373–379, IEEE, 2016.

[3] A. Canciani and J. Raquet, "Absolute positioning using the earth's magnetic anomaly field," *Navigation*, vol. 63, no. 2, pp. 111–126, 2016.

[4] H.-P. Chiu, X. S. Zhou, L. Carlone, F. Dellaert, S. Samarasekera, and R. Kumar, "Constrained optimal selection for multi-sensor robot navigation using plug-and-play factor graphs," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 663–670, 2014.

[5] N. Keivan and G. Sibley, "Online SLAM with Any-time Self-calibration and Automatic Change Detection," *2015 International Conference on Robotics and Automation (ICRA)*, pp. 1–8, 2015.

[6] V. A. Oleshchuk, "Ad-hoc sensor networks: modeling, specification and verification," in *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2003. Proceedings of the Second IEEE International Workshop on*, pp. 76–79, IEEE, 2003.

[7] P. S. Maybeck, *Stochastic Models, Estimation, and Control Volume 1.* Virginia: Navtech, 1982.

[8] W. R. Michalson, "Ensuring gps navigation integrity using receiver autonomous integrity monitoring," *IEEE Aerospace and Electronic Systems Magazine*, vol. 10, no. 10, pp. 31–34, 1995.

[9] B. W. Parkinson and P. Axelrad, "Autonomous gps integrity monitoring using the pseudorange residual," *Navigation*, vol. 35, no. 2, pp. 255–274, 1988.

[10] R. G. Brown and P. McBurney, "Self-contained gps integrity check using maximum solution separation," *Navigation*, vol. 35, no. 1, pp. 41–53, 1988.

[11] T. Walter and P. Enge, "Weighted raim for precision approach," in *PROCEEDINGS OF ION GPS*, vol. 8, pp. 1995–2004, Institute of Navigation, 1995.

[12] E. Wang, M. Cai, and T. Pang, "A simple and effective gps receiver autonomous integrity monitoring and fault isolation approach," in *Control Engineering and Communication Technology (ICCECT), 2012 International Conference on*, pp. 657–660, IEEE, 2012.

[13] S.-Y. R. Young and G. A. McGraw, "Method and system for fault detection and exclusion for multi-sensor navigation systems," May 15 2007. US Patent 7,219,013.

[14] P. Y. Hwang and G. A. McGraw, "Receiver autonomous signal authentication (rasa) based on clock stability analysis," in *Position, Location and Navigation Symposium-PLANS 2014, 2014 IEEE/ION*, pp. 270–281, IEEE, 2014.

[15] M. Joerger and B. Pervan, "Kalman filter-based integrity monitoring against sensor faults," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 2, pp. 349–361, 2013.

[16] L. Gratton, M. Joerger, and B. Pervan, "Carrier phase relative raim algorithms and protection level derivation," *The Journal of Navigation*, vol. 63, no. 2, pp. 215–231, 2010.

[17] M. Joerger, J. Neale, and B. Pervan, "Iridium/gps carrier phase positioning and fault detection over wide areas," in *Proceedings of the 22nd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2009)*, pp. 1371–1385, 2009.

[18] T. Beravs, S. Beguš, J. Podobnik, and M. Munih, "Magnetometer calibration using kalman filter covariance matrix for online estimation of magnetic field orientation," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 8, pp. 2013–2020, 2014.

[19] Z. Wu, Z. Wang, and Y. Ge, "Gravity based online calibration for monolithic triaxial accelerometers' gain and offset drift," in *Intelligent Control and Automation, 2002. Proceedings of the 4th World Congress on*, vol. 3, pp. 2171–2175, IEEE, 2002.

[20] M. Kirkko-Jaakkola, J. Collin, and J. Takala, "Bias prediction for MEMS gyroscopes," *IEEE Sensors Journal*, vol. 12, pp. 2157–2163, June 2012.

[21] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers.," in *Robotics: Science and Systems*, vol. 2, 2013.

[22] H. Miura, T. Yoshida, K. Nakamura, and K. Nakadai, "Slam-based online calibration of asynchronous microphone array for robot audition," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 524–529, IEEE, 2011.

[23] J. D. Jurado and C. C. McGehee, "Complete online algorithm for air data system calibration," *Journal of Aircraft*, pp. 1–12, 2018/10/31 2018.

[24] N. Keivan and G. Sibley, "Constant-time monocular self-calibration," *International Conference on Robotics and Biomimetics*, pp. 1590–1595, 2014.

[25] Z. Yang and S. Shen, "Monocular visual-inertial fusion with online initialization and camera-IMU calibration," *SSRR 2015 - 2015 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2016.

[26] C. Jia and B. L. Evans, "Online calibration and synchronization of cellphone camera and gyroscope," *2013 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013 - Proceedings*, pp. 731–734, 2013.

[27] Y. Ling and S. Shen, "High-precision online markerless stereo extrinsic calibration," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1771–1778, IEEE, oct 2016.

[28] S. Diverdi and J. T. Barron, "Geometric calibration for mobile, stereo, autofocus cameras," *2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016*, 2016.

[29] S. Akhlaghi, N. Zhou, and Z. Huang, "Adaptive adjustment of noise covariance in kalman filter for dynamic state estimation," in *Power & Energy Society General Meeting, 2017 IEEE*, pp. 1–5, IEEE, 2017.

[30] C. H. Bishop, B. J. Etherton, and S. J. Majumdar, "Adaptive sampling with the ensemble transform kalman filter. part i: Theoretical aspects," *Monthly weather review*, vol. 129, no. 3, pp. 420–436, 2001.

[31] J. A. Waller, S. L. Dance, A. S. Lawless, and N. K. Nichols, "Estimating correlated observation error statistics using an ensemble transform kalman filter," *Tellus A: Dynamic Meteorology and Oceanography*, vol. 66, no. 1, p. 23294, 2014.

[32] C.-B. Chang and M. Athans, "Hypothesis testing and state estimation for discrete systems with finite-valued switching parameters," tech. rep., MASSACHUSETTS INST OF TECH CAMBRIDGE ELECTRONIC SYSTEMS LAB, 1977.

[33] P. Eide and P. Maybeck, "An mmae failure detection system for the f-16," *IEEE Transactions on Aerospace and Electronic systems*, vol. 32, no. 3, pp. 1125–1136, 1996.

[34] P. D. Hanlon and P. S. Maybeck, "Multiple-model adaptive estimation using a residual correlation Kalman filter bank," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 36, no. 2, pp. 393–406, 2000.

[35] S. M. Kay, *Fundamentals of Statistical Signal Processing, Vol. II: Detection Theory.* Prentice Hall, 1998.

[36] M. H. Kutner, C. Nachtsheim, and J. Neter, *Applied Linear Regression Models*, vol. 4. McGraw-Hill/Irwin, 2004.

[37] D. M. Bates and D. G. Watts, *Nonlinear regression analysis and its applications.* Wiley series in probability and mathematical statistics. Applied probability and statistics, New York, Chichester: J. Wiley, 1988. Includes indexes.

[38] B. Brumback and M. Srinath, "A chi-square test for fault-detection in kalman filters," *IEEE Transactions on Automatic Control*, vol. 32, no. 6, pp. 552–554, 1987.

[39] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, pp. 716–723, dec 1974.

[40] R. Y. Novoselov, S. M. Herman, S. M. Gadaleta, and A. B. Poore, "Mitigating the effects of residual biases with Schmidt-Kalman filtering," in *2005 7th International Conference on Information Fusion, FUSION*, 2005.

[41] K. M. Brink, "Partial-update schmidt–kalman filter," *Journal of Guidance, Control, and Dynamics*, pp. 1–15, 2017.

[42] K. J. Kauffman, "Scorpion." https://www.afit.edu/docs/Scorpion.pdf, November 2018.

[43] C. F. Van Loan, "Computing integrals involving the matrix exponential," in *IEEE Transactions on Automatic Control*, vol. 23:3, pp. 395–404, June 1978.

[44] P. S. Maybeck, *Stochastic Models, Estimation, and Control Volume 2.* Virginia: Navtech, 1984.