MaRS: A Modular and Robust Sensor-Fusion Framework

Christian Brommer[1], Roland Jung[2], Jan Steinbrener[1], and Stephan Weiss[1]

***Abstract*—State-of-the-art recursive sensor filtering frameworks allow the fusion of multiple sensors tailored to a specific problem but do not allow a dynamic and efficient introduction of additional sensors during runtime - an important feature to enable long-term missions in dynamic environments. This paper presents a robust, modular sensor-fusion framework that enables the addition and removal of sensors at runtime. These sensors could be not *a priori* known to the system. The framework handles the complexity of system and sensor initialization, measurement updates, and switching of asynchronous multi-rate sensor information with sensor self-calibration in a truly modular and generic design. In addition, the framework can handle delayed measurements, out-of-sequence updates, and can monitor sensor health. The introduced *true-modularity* is based on covariance segmentation to allow the isolated (i.e., modular) processing of propagation and updates on a per-sensor basis. We show how crucial properties of the overall state covariance can be maintained as naive implementation of such a modularization would invalidate the covariance matrix. We evaluate our framework for a precision landing scenario switching between combinations of GNSS, barometer, and vision measurements. Tests are performed in simulation and in real-world scenarios to show the validity of the introduced method. The presented framework will be open-sourced and made available online to the community.**

***Index Terms*—Sensor Fusion, State-Estimation, Modularity, Autonomous Navigation**

I. Introduction

STATE-Estimation is an essential part of robotics and engineering. The accurate knowledge of the location of a robotic platform in the world is crucial for navigation, control, and manipulation. Dedicated estimators are repeatedly developed, and most existing approaches are tailored to accomplish a specific task on specific hardware under specific conditions, limiting re-usability if the scenario, sensor suite, or the platform changes. Current open-source and state-of-the-art Extended Kalman Filter (EKF) frameworks start to address this issue, but they are designed to handle a setup of sensors that is pre-defined during the compilation time or start-up phase of the filter. The reference frames of additional sensors are also often pre-defined and are not dynamically adapted to the current situation. Such frameworks do not allow the initialization of sensors during runtime, especially if the sensor definition is not *a priori* known to the system. This limits their application to static hardware configurations and does not support novel applications with modular platforms that can be extended during runtime with sensor modules not known to the core framework (e.g., connectable snake robots or humanoids with exchangeable end-effectors).

A major challenge is that additional sensors require additional calibration states because they are rarely aligned with the robot's estimated body frame, nor are they intrinsically calibrated. Therefore, the number of calibration-states increases with the number of sensors. An increasing number of states requires more operations to perform the estimation (e.g., for propagation and updates in filter-based estimates). The processing time of a naive estimator increases *cubically* $\mathcal{O}(n^3)$ with the number of sensors n due to matrix multiplications. This effect is even worse for delayed and out-of-sequence measurements in a multi-sensor system because delayed signals trigger numerous re-computation steps should the estimator remain credible. Hardware synchronization can mitigate this issue, but it may not always be possible (particularly with dynamic sensor rates). While non-recursive filter formulations (e.g., graph optimization-based) have been shown to be able to initialize previously unknown sensors during runtime, their computational load makes them ill-suited for execution on resource-constrained platforms such as Unmanned

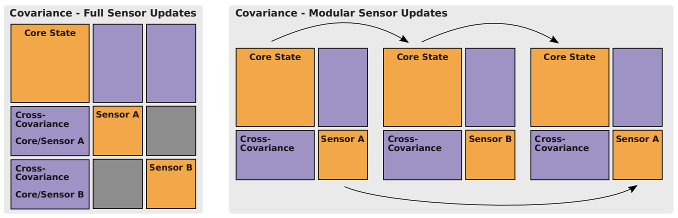

Fig. 1. True modularity. Left: Structure of a state-of-the-art multi-sensor fusion approach. The covariance matrix is always fully updated. Right: Modular segmentation of the update and propagation process. Only the core and currently active sensor state covariance is used to perform the update.

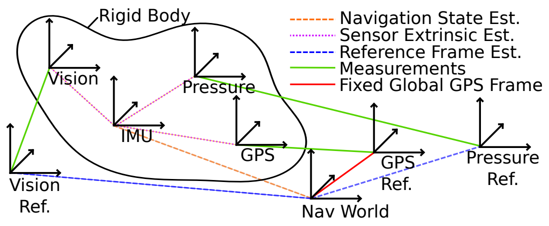

Fig. 2. Estimated state variables with self-calibration are shown as dotted lines; measurements and one fixed global reference frame (e.g. Global Navigation Satellite System (GNSS)) are shown as continuous lines.

Manuscript received: August, 11, 2020; Revised October, 10, 2020; Accepted September, 28, 2020.

This paper was recommended for publication by Editor Dan Popa upon evaluation of the Associate Editor and Reviewers' comments. The research leading to these results was supported by the ARL within the BAA W911NF-12-R-0011 under grant agreement W911NF-16-2-0112, the University of Klagenfurt within the doctoral school KPK-NAV, and the European Commission under grant agreement 871260 - BugWright2.

[1] Control of Networked Systems Group, University of Klagenfurt, Austria. E-Mail: { christian.brommer, jan.steinbrener, stephan.weiss }@ieee.org

[2]Karl Popper School on Networked Autonomous Aerial Vehicles, University of Klagenfurt, Austria. E-Mail: roland.jung@ieee.org

Digital Object Identifier (DOI): see top of this page.

Aerial Vehicles (UAVs). Here, we present a recursive, unified, and modular multi-sensor-fusion framework with support for efficient, multiple asynchronous updates resulting in constant complexity independent of the number of sensors. It further provides generalized interfaces that allow an easy exchange of components between projects and collaborators. The contributions of this work are the following:

- The design and implementation of a truly modular multi-sensor fusion framework as a recursive filter with the capability of *on-line* addition of previously undefined sensors with delay and asynchronous measurements. These sensors will be self-initialized and self-calibrated based on their extrinsic states, which are added to the system.
- A novel approach for correct covariance segmentation, which preserves the properties of a covariance matrix throughout the isolated processing of individually joined segments. This renders the framework both consistent and computationally tractable on constrained platforms: the complexity depends only linearly on the number of sensors and the propagation step constant/independent of the number of sensors.
- Statistically relevant tests in simulations and verification of the proposed framework with real data.

On-line sensor addition is achieved by decoupling the navigation states (e.g., position, velocity, and attitude for a mobile system) from calibration states of individual sensors (e.g., the transformation between sensor and agent body frame). This allows the introduction of a *sensor-update-module* during runtime (e.g., through independently launchable nodelets in a Robot Operating System (ROS) environment). Our design also accounts for offsets between global and local references maintaining smooth state evolution upon inclusion of a new reference frame. To maintain consistency despite this modularization, we introduce a covariance segmentation approach, which retains the filter's credibility, to correctly propagate isolated covariance components by maintaining fundamental properties of covariance matrices. This is crucial for the correct application of propagation and update steps. A naive approach would result in ill-conditioned covariances. The presented approach reduces the complexity of updates and renders the processing time of the propagation as well as the update phase constant and independent of the number of sensors.

The approach provides continuous self-calibration (see Fig. 2) while keeping maximum flexibility at a low computational cost. We validate the proper propagation of information (i.e., observability properties), the credibility of the overall approach, and the performance with a statistically relevant number of simulations. We illustrate the feasibility of our approach for computationally-constrained platforms with real-world experiments and an UAV. The experiments are performed with Inertial Measurement Unit (IMU) driven dynamics; however, different dynamic formulations are possible, and the framework can, of course, be deployed on other platforms, not limiting the contribution to UAVs.

II. Related Work

State estimation with pre-defined sensor suites and complementing calibration states, including self-calibration and delay compensation with multi-sensor rates, have been studied thoroughly in literature. The Single Sensor Fusion framework (SSF) presented by [1] covers the topics of online self-calibration and accurate handling of sensor delays (out-of-sequence updates). An extended version of SSF was used by [2] in a multi-sensor setup for long-duration autonomy. The Multi-Sensor Fusion framework (MSF) has been introduced by [3], and [4] has presented similar work that details relative and absolute sensor updates using local vision updates and global position information as an example. While both frameworks, SSF, and MSF can accommodate sensor outages, the work of [5] extended the MSF framework and studied the topic of online sensor initialization and switching based on sensor availability and health metrics.

[6] introduced a method for the handling of delayed measurements, designed for computationally-constrained embedded systems. [7] presented a generalized extended Kalman filter implementation based on ROS. This framework defines its sensor structure during startup but does not allow modification of the setup during runtime. Sensor measurements are assumed to be expressed in the robot's origin. The framework does not introduce sensor calibration states and does not perform online self-calibration. It neither estimates Gyroscopic biases for the IMU, and the process noise of the system is tuned by hand. The method presented here goes further and allows the incorporation and removal of sensors that are not *a priori* known to the system during runtime by decoupling the core states from the sensor states. This allows a decentralized yet tightly-coupled processing of sensor information.

The work of [8] describes the state-of-the-art centralized and decentralized sensor-fusion for driver-assistance systems and discusses the current challenges of this approach. In short, centralized approaches allow tightly-coupled estimation but require high communication bandwidth. Existing approaches are also hard to extend and require extensive workload for the implementation of new sensor instances. State-of-the-art decentralized systems make use of loosely-coupled sensor integration, which has the disadvantage of inconsistencies because of inadequate handling of the sensor and core states cross-covariances. The focus of the presented work relies specifically on modularity and consistency/credibility; however, it will also benefit the development of decentralized systems (e.g., swarms). It allows tightly-coupled sensor-fusion with reduced bandwidth between system components because the states of a sensor instance can be stored and processed locally. It further simplifies the development and extension of systems by minimizing the workload for integrating new sensors and allows online retrofitting.

The work of [9] and [10] studied the modularization of multi-sensor fusion and presented a vector graph-based method which employs a real-time batch optimization process. Both authors' work focuses on the optimal and the minimal selection as a subset of the given sensor suit and covers observability for sensor selection. The authors perform plug and play experiments by abstracting the sensor to avoid the direct use of physical measurements. It is important to note that the use of vector graph-based methods is limited in terms of scalability, especially in combination with computationally-constrained resources. The presented work is different because it focuses on a truly modular approach with a recursive filtering technique. The presented approach minimizes the size of covariance matrices, reducing the number of mathematical operations and increasing performance/scalability. To the best of our knowledge, no truly modular recursive filter approach, as presented in this paper, has been reported in the literature.

III. Method

A. Truly Modular Sensor Fusion

Recursive filters such as EKFs require all states and covariances during the update and propagation phase. A typical setup of a filter for estimating a system, defines core states that describe the essential variables of a platform that are necessary for propagation and to perform controls. We use the core state definition established by [1] and shown by Equation (1). The essential core states are the translation from the world to the IMU/body frame ${}_W\mathbf{p}_{WI} \equiv \mathbf{p}_{WI}$ expressed in the world frame, velocity $\mathbf{v}_{WI}$, the orientation of the IMU in the world frame $\mathbf{q}_{WI}$, gyroscopic bias $\mathbf{b}_\omega$ and accelerometer bias $\mathbf{b}_a$, with ${}_C\mathbf{p}_{AB} = \mathbf{R}_{(\mathbf{q}_{CA})}\,{}_A\mathbf{p}_{AB}$ and $\mathbf{R}_{(\mathbf{q}_{CA})} \equiv \mathbf{R}_{CA}$.

$$\mathbf{X}_\mathsf{C} = \left[\mathbf{p}_{WI}^\mathsf{T}, \mathbf{v}_{WI}^\mathsf{T}, \mathbf{q}_{WI}^\mathsf{T}, \mathbf{b}_\omega^\mathsf{T}, \mathbf{b}_a^\mathsf{T}\right]^\mathsf{T} \tag{1}$$

Generally, for mobile systems, the state and covariance can be propagated by an IMU driven and time-dependent dynamic model. The following differential equations govern the state-dynamics, with $\Omega(\boldsymbol{\omega})$ being the right side Quaternion multiplication matrix for $\boldsymbol{\omega}$, gravity expressed in the world frame $\mathbf{g}$, and $\mathbf{n}_{\mathbf{b}_a}$, $\mathbf{n}_{\mathbf{b}_\omega}$ being zero mean white Gaussian noise of the accelerometer and gyroscope measurements.

$$\dot{\mathbf{p}}_{WI} = \mathbf{v}_{WI} \tag{2}$$
$$\dot{\mathbf{v}}_{WI} = \mathbf{R}_{(\mathbf{q}_{WI})}(\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a) - \mathbf{g} \tag{3}$$
$$\dot{\mathbf{q}}_{WI} = \frac{1}{2}\Omega(\boldsymbol{\omega}_m - \mathbf{b}_w - \mathbf{n}_w)\mathbf{q}_{WI} \tag{4}$$
$$\dot{\mathbf{b}}_\omega = \mathbf{n}_{\mathbf{b}_\omega},\ \dot{\mathbf{b}}_a = \mathbf{n}_{\mathbf{b}_a}. \tag{5}$$

If the system provides additional sensors, they are likely not aligned with the center of the platform. The extrinsics of individual sensors can be implemented as calibration states and may be estimated online. Given a system with e.g., two additional sensors S_1 and S_2 the core state can be augmented with their extrinsic calibration states accordingly

$$\mathbf{X} = \left[\mathbf{X}_\mathsf{C};\ \mathbf{X}_{\mathsf{S}_1};\ \mathbf{X}_{\mathsf{S}_2}\right]. \tag{6}$$

The observation of additional sensors introduces cross-correlations between the core and sensor states, resulting in cross-covariances in the covariance matrix $\mathbf{P}$. The joint covariance matrix after sensor observations is

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_\mathsf{C} & \mathbf{P}_{\mathsf{CS}_1} & \mathbf{P}_{\mathsf{CS}_2} \\ \mathbf{P}_{\mathsf{S}_1\mathsf{C}} & \mathbf{P}_{\mathsf{S}_1} & 0 \\ \mathbf{P}_{\mathsf{S}_2\mathsf{C}} & 0 & \mathbf{P}_{\mathsf{S}_2} \end{bmatrix}, \tag{7}$$

with $\mathbf{P}_{\mathsf{CS}_2} = (\mathbf{P}_{\mathsf{S}_2\mathsf{C}})^\mathsf{T}$ and sensors S_1 and S_2 assumed to be independent to each other.

Starting from this structure, we propose a segmentation approach that isolates the core and sensor covariance components. Performing the propagation on the isolated core states reduces the size of the covariance matrix, minimizing computational effort. A possible filtering routine with covariance segmentation for a two-sensor scenario is shown by Figure 3.

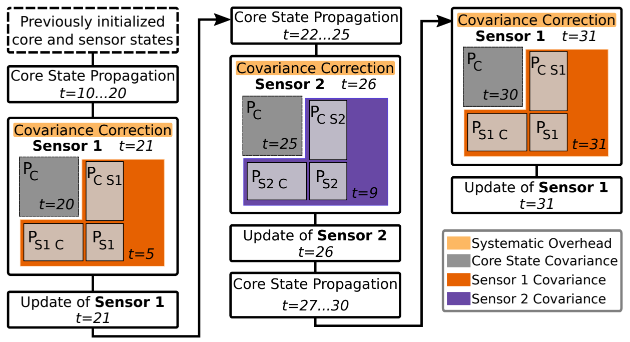
Previously initialized core and sensor states
Core State Propagation t=10...20
Covariance Correction Sensor 1 t=21
P_C t=20
P_C S1
P_S1 C
P_S1
t=5
Update of Sensor 1 t=21
Core State Propagation t=22...25
Covariance Correction Sensor 2 t=26
P_C t=25
P_C S2
P_S2 C
P_S2
t=9
Update of Sensor 2 t=26
Core State Propagation t=27...30
Covariance Correction Sensor 1 t=31
P_C t=30
P_C S1
P_S1 C
P_S1
t=31
Update of Sensor 1 t=31
Systematic Overhead
Core State Covariance
Sensor 1 Covariance
Sensor 2 Covariance

Fig. 3. The figure shows a representative sequence for the truly modular filter process with two sensors. The covariance correction element performs two steps, first the generation of the state-transition blocks for the propagation of the sensor covariance, and second, the Eigenvalue covariance correction. Please note that the sensor covariance and cross-covariance (orange and purple boxes) are stored at the time of their update and do not evolve over time until their next update. Sensor measurements occur at $t = 21, 26,$ and 31s.

The scenario describes the filtering procedure for a time section between t=0 s and 31 s. The filter is initialized at t=0 s, two sensors have been added and initialized during runtime, S_1 at t=5 s, and S_2 at t=9 s. The covariance handling is performed as follows: The core covariance and states are propagated separately in the time t=10 s and 20 s. Sensors S_1 and S_2 have not been part of this propagation. S_1 provides a measurement at t=21 s; at this point, the latest sensor covariance $\mathbf{P}_{\mathsf{S}1}$ and cross-covariance with the core states $\mathbf{P}_{\mathsf{CS}1}$ at t=5 s (orange segment) is joined with the latest core state covariance $\mathbf{P}_\mathsf{C}$ at t=20 s, and the update for t=21 s is performed. The sensor covariance $\mathbf{P}_{\mathsf{S}1}$ and sensor/core cross-covariance $\mathbf{P}_{\mathsf{CS}1}$ is separated from the core covariance $\mathbf{P}_\mathsf{C}$ afterwards and stored until the next update of S_1 at t=31 s.

The core state is propagated until the measurement update of S_2 at t=26 s, and the latest covariance segments $\mathbf{P}_\mathsf{C}$ of the core (t=25 s) and S_2 (t=9 s) are used for the update at t=26 s. At this point, the process continues with the same procedures: the propagation of the isolated core covariance $\mathbf{P}_\mathsf{C}$ and the update of individual sensor states with the core. One important aspect is that measurement updates of one sensor are separated from the state of any other sensor (see Eq. (7)). This is one of the key components of the introduced covariance segmentation that allows true modularity. The routine shows that any sensor can be added or removed without interfering with other sensor covariances. We call this *truly modular* since only the minimal representation of the current state, and covariance segments are joined for a particular update or propagation.

B. Consistent Truly Modular Covariance Estimation

The described covariance segmentation introduces two problems. Due to this approach, two or more sensors are never updated in the same step, and thus, no cross-covariance terms between sensors are generated. We make the design choice that the cross-covariance between any sensor states are zero (see Eq. (7)). Thus, all additional auxiliary states of individual sensors are independent, but the covariance of an individual sensor and its cross-covariance with the core state is maintained (see Fig. 1). An intuitive physical example can be given by using a 3DoF magnetometer and 3DoF GNSS sensor. The rotational calibration of the magnetometer with respect to the IMU and the translation of the GNSS with respect to the IMU do not have a physical relation. Although these cross-covariance do exist from an analytical point of view; they are negligible as the experiments in Section IV-A validate. Thus, negligible losses in accuracy allow vast performance improvements given the gained recursive modularity.

The second problem is the validity of the covariance matrix for the joined covariance segments. The covariance segments were calculated for different points in time and do not include the same amount of sample data, which leads to non-positive-semidefiniteness. Thus, we propose pre-update routines to reintroduce the information that was not handled during the propagation and individual update phases. We then select the closest valid (positive-semidefinite) covariance matrix from this augmented matrix.

1) Propagation: The information that each isolated sensor component was missing during the propagation phase can be fetched and propagated forward consistently to the current update step. In [11] it is shown that the cross-covariances can be independently propagated using the *state-transition matrix series*. The state-transition matrix series $\mathbf{\Phi}(m,n)$ between two time instances $\mathrm{t}(m)$ and $\mathrm{t}(n)$ is defined as

$$\mathbf{\Phi}(m,n) = \mathbf{\Phi}_n \mathbf{\Phi}_{n-1} \ \ldots \ \mathbf{\Phi}_m \text{ with } \mathrm{t}(m) < \mathrm{t}(n), \tag{8}$$

with $\mathbf{\Phi}_k$ as the discrete state-transition matrix $\mathbf{\Phi}_{k|k-1}$ that encodes the state dynamic, evaluated based on the system input, and integrated for the propagation step $\delta\mathrm{t} = \mathrm{t}(k) - \mathrm{t}(k-1)$. The corresponding cross-covariance $\mathbf{P}_{\mathsf{CS}}$ between core C and sensor S can be propagated from the time instance $\mathrm{t}(m)$ until $\mathrm{t}(n)$ with

$$\mathbf{P}_{\mathsf{CS},n(-)} = \mathbf{\Phi}_{\mathsf{C}}(m,n)\mathbf{P}_{\mathsf{CS},m}\mathbf{\Phi}_{\mathsf{S}}(m,n)^{\mathsf{T}}, \tag{9}$$

$\mathbf{\Phi}_{\mathsf{C}}(m,n)$ being the state-transition matrix series of the core and $\mathbf{\Phi}_{\mathsf{S}}(m,n)$ for the sensor state. Storing a history of state-transition matrices, allows the generation of a state-transition matrix series to propagate sensor covariance and cross-covariance between core and sensor states. The sensor covariance $\mathbf{P}_{\mathsf{CS}}$ inherits the information that was not introduced while the core $\mathbf{P}_{\mathsf{C}}$ was propagated in isolation. This *on-demand* information inheritance allows to only compute the core states at each propagation step, keeping this step at constant complexity independent of the number of sensors, but requires a computational spike for the pre-update step.

2) Updates: [11] also showed that indirect observations affect core and sensor states because of cross-correlations between the core and individual sensor states. This means that a sensor observation, e.g., provided by S_1, results in an update and correction of states correlated with the core state e.g., those of S_2. Due to this, sensor covariances can usually not be removed and reintroduced directly. Considering Figure 3, the removal of a previously introduced sensor S_1 at t=21 s and its reintroduction at t=31 s after other measurement updates have been performed (S_2 at t=26 s), invalidates the covariance matrix, which becomes non-positive semidefinite and is called a pseudo covariance matrix. This non-continuous evolution of the segmented covariance matrix can be corrected by enforcing the required properties of covariance and correlation matrices, respectively. Covariance matrices are symmetric and positive-semidefinite $\mathbf{P} \in S^n_+$, which ensures that its correlations are coherent, but it is not guaranteed that the combination of covariance segments, as described above, satisfies this property. As an example: Given the covariance matrix $\mathbf{P}$ in Equation (10): Let $\mathbf{P}_{AB}$ and $\mathbf{P}_{BC}$ be a positive cross-covariance between the states. Due to this relation, $\mathbf{P}_{AC}$ needs to represent a positive correlation as well.

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_A & \mathbf{P}_{AB} & \mathbf{P}_{AC} \\ \mathbf{P}_{BA} & \mathbf{P}_B & \mathbf{P}_{BC} \\ \mathbf{P}_{CA} & \mathbf{P}_{CB} & \mathbf{P}_C \end{bmatrix} \in S^n_+ \tag{10}$$

A covariance correction step needs to be applied to accommodate this issue. [12] and [13] discuss a variety of methods to estimate the nearest positive-semidefinite covariance matrix of a given pseudo covariance matrix. The more interesting approaches are the Eigenvalue method and the Scaling/Hypersphere decomposition with angular parametrization, which have not been applied to the field of state-estimation to our knowledge.

The scaling method uses an optimization process to minimize the Frobenius distance (lower caps are the scalar elements of a matrix) $\mathbf{A} = \sum_i^n \sum_j^n (p_{i,j} - \tilde{p}_{i,j})^2$ with respect to a given covariance matrix where $\mathbf{P}$ is the true covariance matrix and $\tilde{\mathbf{P}}$ is the closest approximation. The Eigenvalue method approximates a positive-semidefinite matrix by correcting negative Eigenvalues. [14] proves that the Eigenvalue method also minimizes the Frobenius norm. Due to its deterministic nature and the lower complexity, the Eigenvalue method is the preferred choice for the presented real-time estimation problem. To perform the Eigenvalue correction, the first step is to decompose the covariance matrix $\mathbf{P} = \mathbf{DED}^T$ that needs to be adapted. $\mathbf{E}$ is a diagonal matrix with Eigenvalues, and $\mathbf{D}$ are the Eigenvectors. If the covariance matrix is non-positive-semidefinite, then a subset of the Eigenvalues $\mathbf{E}_{(<0)}$ is negative. These can be corrected by performing the:

- Absolute Eigenvalue correction (ABS), to preserve the dimension that is spanned by the Eigenvectors.
- Zero Eigenvalue correction (Zero), performing the minimal change required to gain a positive determinant, and
- Delta Eigenvalue correction (Delta), which sets the negative Eigenvalues to a positive empirical parameter.

The covariance matrix is then constructed based on the corrected Eigenvalues and can be used to update the recursive filter. The three methods are applied for the framework and evaluated in Section IV-A.

C. Implementation

The framework is structured in logical blocks (see Fig. 4) that represent the system design. Each component is self-contained with clearly defined interfaces for exchangeability. The core logic handles the organizational part of the framework and is the bridge between the buffer (see Fig. 5) and all sensor components. Its high-level logic determines if measurements are still valuable to the system or rejected (e.g. if the measurement was delayed and is older than the latest buffered

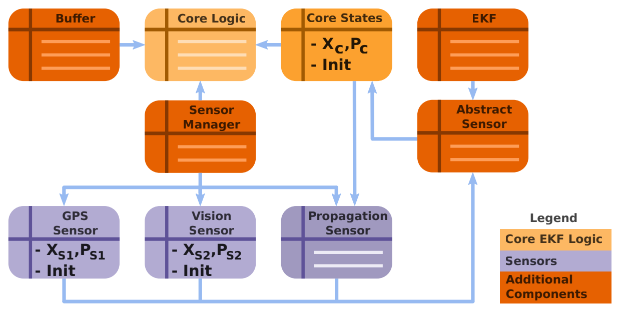

Fig. 4. Modular system design.

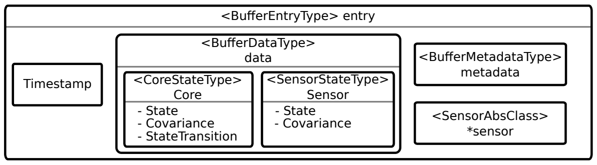

Fig. 5. Data structure of a single buffer entry.

entry). The core logic redirects dynamics measurements (e.g., from an IMU) to the core state module, which propagates the core state vector and its covariance. If measurements are associated with any other sensor instance (e.g., GNSS), then the core logic requests the latest state entry from the buffer and all state-transition matrices starting at the previous update of this sensor until the current step. The information is used to generate the state-transition matrix series described in Section III-B, which is used to propagate the cross-covariance terms of the core and sensor states. The propagated sensor cross-covariance is used to build a covariance matrix with the corresponding sensor and core covariance. The resulting covariance matrix is corrected with the Eigenvalue method and is passed to the sensor instance, which performs the update. The sensor instance can handle the reduced covariance matrix and state-prior (latest core and sensor state) as it is done for the classical approach. This also allows the use of statistical tests (e.g. χ^2 test) within the sensor's update step. The sensor instance returns the updated states and covariance segments of the core and the sensor to the core logic, which stores it in the buffer. The presented method renders the core agnostic to the sensor definition, which allows the arbitrary addition of sensors. Each sensor instance is self-contained, performs its own updates, and applies the corrections to its states. The same holds if a new sensor instance is added during runtime. A sensor module also handles its initialization based on the current core state, provided by the core logic. The framework is programmed in Matlab for fast prototyping, and implemented in C++ for high-performance applications. The C++ framework has minimal dependencies and only relies on the Eigen library. A ROS package that uses the API of the C++ library is also provided.

IV. Experiments

A. Validity and Observability

Due to the assumptions made in Section III-B, the approach needs to be evaluated in terms of performance and characteristics in simulation and the real-world. The tests of this section have three objectives:

1) The evaluation of the three Eigenvalue correction methods (ABS, Zero, Delta = 0.05).
2) An experimental analysis that unobservable vision states become observable by introducing a global pose sensor (i.e., that correct/consistent information is propagated despite the simplifications).
3) The validation of the overall modular approach.

The setup is as follows: we use the same simulated ground-truth trajectory to generate 20 independent Monte-Carlo datasets, which allow a statistically significant number of repetitions. The trajectory has a duration of 15 minutes with continuously varying velocity, accelerations, and randomly introduced smooth orientations. Each sequence has the same trajectory and the same Gaussian noise characteristics for sensor and IMU measurements. The datasets provide 200 Hz IMU measurements for propagation, 6DoF loosely-coupled vision pose (10 Hz), and 6DoF pose sensor measurements (50 Hz). The validity of the filter and the underlying modular approach is quantified by the:

- Average Normalized Estimation Error Squared (ANEES) described by [15] to determine the filter characteristics in terms of consistency and credibility,
- State error plots for time-dependent coherence, and
- Root-Mean-Square Error (RMSE) w.r.t. ground-truth, comparing the classical filter and our modular approach.

One dataset was processed with the classical full filter approach to establish a baseline for the best-case scenario (similar to the framework introduced by [3]). Each of the 20 datasets was processed with the modular filter definition using the three different Eigenvalue correction methods. The individual result of each state, from the modular approach, was used to generate an RMSE with respect to the full filter scenario. The mean of the individual core state RMSE for the different Eigenvalue correction methods are shown in Figure 6. The results show that the absolute Eigenvalue correction method performs best for all states except for velocity, where the zero method performs slightly better on two out of the three dimensions.

The same test was performed for the observability validation with introduced random state initializations for each sequence. The calibration of the vision-world reference frame for the vision sensor is unobservable but can be rendered observable by introducing a global pose sensor. Since we are using the segmentation of the covariance matrix, the vision and pose sensor are never jointly present in the covariance matrix in the same update step. Thus, we need to validate that the usual flow of information from the pose sensor to the vision sensor, which contributes to the observability of the vision-world reference frame due to cross-covariances, can be recovered from the core

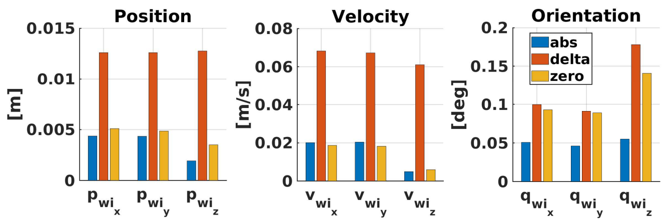

Fig. 6. We generated results with the classical full filter approach and 20 datasets each, with the three Eigenvalue correction methods of the modular approach. The graph shows the mean of the RMSE between the results of the classical and the modular filter for the essential core states.

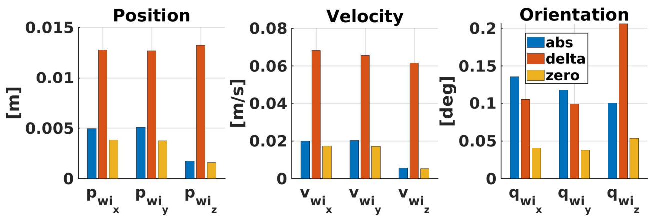

Fig. 7. We applied randomly generated and minorly wrong initializations of the states that are unobservable without using other sensor modalities to prove that the method preserves observability properties when using multiple sensors. The initial covariance encloses the error of the initialization by 3σ to allow for correct convergence.

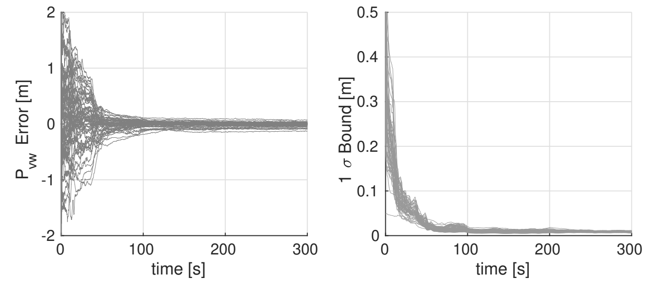

Fig. 8. The vision state $\mathbf{p}_{VW}$ is unobservable without additional global information. The plot shows the convergence of the state error (left) and standard deviation (right) using the modular approach with the absolute Eigenvalue method, 20 Monte-Carlo independent datasets, and varying errors on the initialization of $\mathbf{p}_{VW}$. The result further proves that observability properties are preserved with the modular approach.

states despite the covariance segmentation for the modular approach. This is not inherently given as we explicitly decouple the covariances of the sensors. If this information flow is not maintained, our approach would not be adequate for practical usage. For testing purposes, the state initializations for the vision sensor are altered for each dataset, and the covariance is adapted such that the introduced error is enclosed by a 3σ bound.

The RMSE of the core state is expected to be significantly higher if the vision states do not converge. Figures 6 and 7 as well as Table II confirm that the Eigenvalue correction to a small delta value ($\Delta = 0.05$) shows the least accurate performance, and motivate the usage of the absolute Eigenvalue method, which was therefore used for the remaining experiments and in Section IV. The low RMSE for the described scenario, shown by Figure 7, and the correct convergence of the state error and covariance in Figure 8, using the absolute Eigenvalue method, confirm that the modular approach preserves observability properties.

The next step is the validation of the overall filter credibility. We are using a sensor setup with two pose sensors for this test. The set of 20 datasets is processed with the modular and full filter setup, and the NEES for each run is used to generate the ANEES. Figure 9 shows the ANEES for the full and the modular approach with their corresponding mean. It also shows the 3σ upper-bound of the ANEES based on the number of states and datasets. Both ANEES results are below the upper 3σ ANEES bound, and the individual mean of the ANEES is shown in Table II.

The error plot for this scenario is not shown because the state errors are small and well presented by the RMSE in Table I. Although the mean error is slightly higher due to our approximation, the credibility is still given. The same

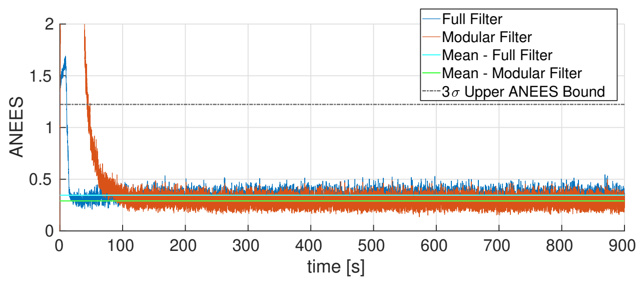

Fig. 9. ANEES for the core states of the full (blue) and modular (red) filter implementation using the absolute Eigenvalue method. We used 20 Monte-Carlo independent datasets to generate a statistically significant characterization of both filter methods. The plot shows the mean for both ANEES after the individual filter method converged. The upper 3σ bound of the the ANEES represented by the dashed line, is based on the number of core states and datasets.

evaluation is done for real-world in-flight data with additional environmental effects such as vibrations of the rotors that affect IMU readings in Section IV-C.

TABLE I
STATE ERRORS FOR THE FULL AND MODULAR FILTER DEFINITION

	$\mathbf{p}_{WI}$ [cm]			$\mathbf{q}_{WI}$ [degree]		
	x	y	z	roll	pitch	yaw
Full μ	0.84	0.83	1.27	0.361	0.375	1.707
Full σ	0.08	0.12	0.48	0.040	0.082	0.332
Modular μ	1.51	1.60	1.42	0.509	0.478	1.154
Modular σ	0.29	0.46	0.40	0.125	0.084	0.741

TABLE II
SUMMARY OF THE MEAN FOR THE ANEES RESULTS

States	Full Filter	Modular Abs	Modular Zero
Nav. Core	0.35	0.3	0.3
Pose Sensor	0.9	0.6	13

B. Performance

The performance of the modular filter is another essential aspect. This section presents timing profiles for a standard scenario and a scenario that forces the framework to repropagate states because of a delayed sensor measurement. Timing profiles are generated with three complete runs for each data point. Figure 10 shows the processing time of the update and propagation step for a series of 1-10 pose sensors. Each sensor introduces a 6DoF calibration state for translation and orientation. The core error state is defined with 15 states, derived from Section III-C and [1]. Thus, the figure shows the timings for 'one sensor' + 'core state' (21 States) and 'ten sensors' + 'core states' (75 states). Considering the case with 75 states: The corresponding covariance matrix has 5625 elements, which are processed by the classical approach for each update and propagation step. The benefit of the modular version is that it only processes the core state during the propagation phase and the core state with one additional sensor ($21 \cdot 21 = 441$) during any update phase.

The evaluation confirms that the propagation (Fig. 10, right) for the modular approach is independent of the number of additional sensors while the processing time of the classical implementation increases with the number of additional states. The processing time of the update (Fig. 10, left) for the classical approach grows exponentially while the modular approach grows linearly. The modular version is more efficient in terms of the total processing time (update + propagation phase), starting at a scenario with three additional sensors (see

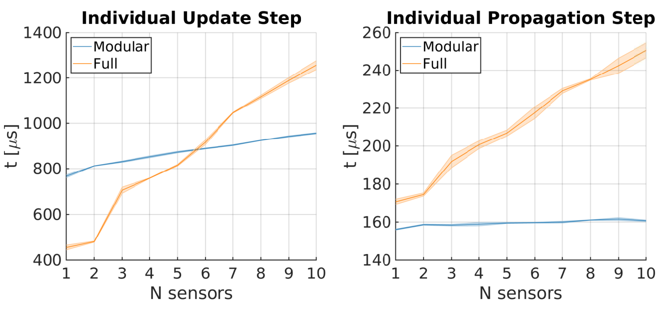

Fig. 10. Timing profile for the classical full and modular EKF approach. The time includes the buffer access and communication of data between the filter instances. This is done to provide a fair comparison since the modular approach introduces a slight overhead with respect to the classical approach. Overall, the modular version still outperforms the classical approach.

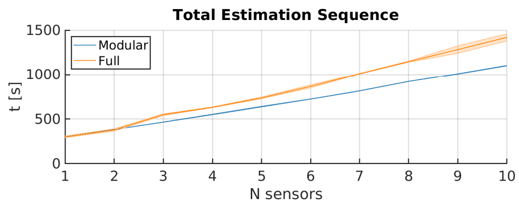

Fig. 11. Total running time comparing the full and modular method over a range of active sensors.

Fig. 11). The reason is a slight overhead due to the covariance correction, being the state transition block generation and the Eigenvalue correction, shown by the flow chart in Figure 3. The overall efficiency of the whole system is higher for the modular approach due to the decreased sensor update, and core propagation time.

C. Vision Aided Landing Scenario with Sensor Switching

This section presents experiments with a realistic flight scenario that is performed in simulation (see Fig. 13) and the real-world (see Fig. 16). The setup uses a GNSS sensor that provides position and velocity measurements at 5 Hz with a position standard deviation of $\sigma_{\mathbf{p}_g} = [0.85\ 0.85\ 2.16]^\mathsf{T}$ according to [16] and $\sigma_{\mathbf{v}_g} = 0.15\,\mathrm{m/s}^2$ for the velocity measurement as suggested by [17]. The setup also includes a loosely-coupled vision sensor in the form of a RealSense T265 with $\sigma_{\mathbf{p}_v} = 0.05\,\mathrm{m}$ for the position and $\sigma_{\mathbf{R}_v} = 1^\circ$ for the orientation measurement. The sensor suite further includes an NXP MPXH6115A integrated pressure sensor with $\sigma_{\mathbf{p}_p} = 0.15\,\mathrm{m}$. Sensor delays are not intentionally introduced. Figure 12 shows the flight profile and phases in which the sensors are switched with the same self-calibration states that are shown by Figure 2. Sensor states are initialized based on the current core state, and the covariance is initialized to enclose the possible error by a 3σ bound. The experiment is performed with 0.5 m/s velocity for all translations. The vehicle performs a vision based takeoff until an altitude of 3 m is reached (segment ①). The GNSS and barometric sensors are initialized in ②. Since these two sensor instances are not *a priori* known to the system, this event represents the addition of new sensors to the system. The vision sensor is deactivated at some point after the start of the horizontal translation. The vehicle performs a 3 m translation in the x-direction, holds at ④, translates 1 m in the y-direction, and returns to the takeoff location ③. Back in ②, the vision sensor is initialized to the current location, and after a short overlapping period,

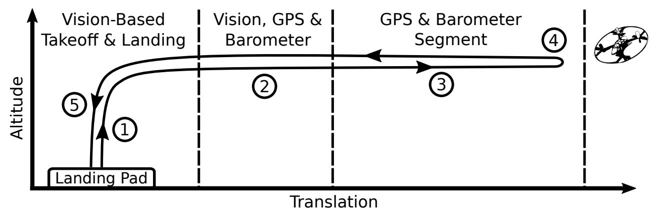

Fig. 12. Experiment THL flight profile with sensor switching cues.

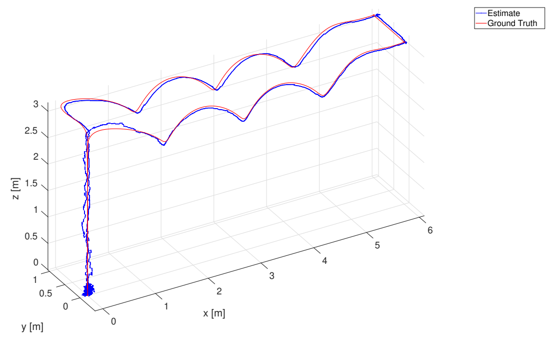

Fig. 13. 3D position estimate (blue) of the simulated sensor switching scenario and overlayed ground-truth (red). RMSE are shown by Table III. The curvy path allows for improved yaw estimation using GNSS and pressure sensors.

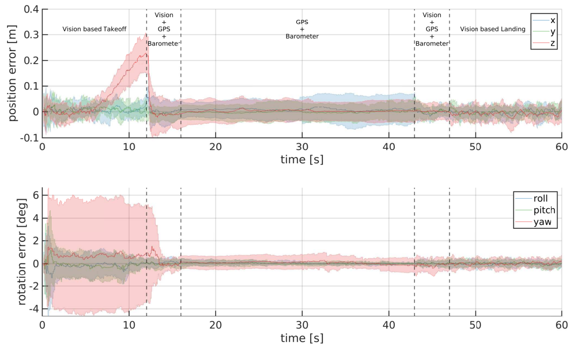

Fig. 14. State error for the position and orientation of the core state. This scenario was performed with 20 datasets to gain a statistically significant result for the truly modular approach. The initial increase of the error in z-position is caused by vision drift due to the takeoff maneuver, which also causes the increased initial covariance of the rotation in yaw.

the GNSS and barometric sensors are deactivated. The vehicle performs a vision-based landing at ⑤.

The real-world experiment is performed in a motion capturing room that provides 6DoF ground-truth for the vehicle's pose. The real vision and pressure measurements are used, and the GNSS position and velocity measurements are generated based on the ground-truth, with normal distributed noise, according to the characteristics mentioned before. The simulated and real-world scenarios do not provide synchronized measurements, and the datasets have high acceleration sections to render the bias of the core state observable. The presented modular state estimation framework performs self-initialization and self-calibration of the individual sensor reference frames and extrinsics based on the current state and sensor measurement.

The results of the simulation (see Fig. 14, Fig. 15, and Table III) further confirm the validity of the approach. They show that the covariance converges quickly after the absolute measurement is introduced and is consistent but underconfident throughout the experiment shown by the ANEES plot. The low RMSE in Table III also confirms the validity of the approach. The results of the real-world scenario (see Fig. 16

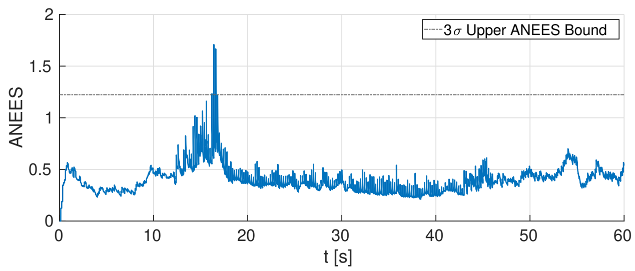

Fig. 15. ANEES for the simulated sensor switching scenario.

TABLE III
RMSE FOR THE SIMULATED AND REAL-WORLD THL SCENARIO.

	$\mathbf{p}_{WI}$ [cm] x	y	z	$\mathbf{q}_{WI}$ [deg] roll	pitch	yaw
Simulated						
Modular μ	4.03	3.27	6.23	0.62	0.65	1.94
Modular σ	1.17	0.61	1.25	0.156	0.199	1.083
Real-World						
Modular μ	15.23	12.65	15.06	2.59	1.97	2.20
Modular σ	14.24	11.54	13.03	2.51	1.73	1.49

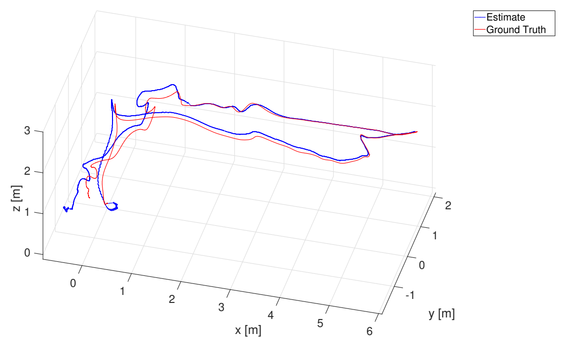

Fig. 16. 3D estimate of real-world data with sensor switching (blue) and overlayed ground-truth (red).

and Table III) illustrate that similar results are obtained with real data.

V. Conclusions

We introduce a novel truly modular multi-sensor fusion approach based on state covariance segmentation, which allows for the addition and removal of sensors at runtime with a significant gain of performance. Naive separation of covariance elements and successive propagation and update steps invalidates the fundamental properties of a covariance matrix. The introduced approach preserves these properties and ensures a consistent filter process. Extensive experiments in simulation and real-world prove that the true modularity approach is credible based on statistically significant ANEES analysis. Furthermore, the modular approach preserves observability, performs self-calibration, and self-initialization. This was shown throughout a vision based takeoff, transition, and landing scenario with four different sensor measurement updates. The scenario showed that the filter remains stable, consistent, and accurate throughout the presented sensor switching scenario with four sensor switching cues and two self-initialization procedures. All scenarios have been performed with asynchronous sensor measurements both in simulation and in a real flight. The modular approach outperforms the classical approach due to faster sensor updates, which improves general scalability for implementations that use this approach.

It was also shown that the propagation phase of the modular approach is constant and invariant to the number of sensors, while the processing time of the classical approach grows exponentially. This is especially interesting if a system uses sensors that introduce measurement delays because the modular approach requires significantly less time to perform a repropagation step.

References

[1] S. Weiss and R. Y. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 231855, pp. 4531–4537, 2011.

[2] C. Brommer, D. Malyuta, D. Hentzen, and R. Brockers, "Long-duration autonomy for small rotorcraft UAS including recharging," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2018, pp. 7252–7258.

[3] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Nov. 2013.

[4] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4974–4981, 2014.

[5] K. Hausman, S. Weiss, R. Brockers, L. Matthies, and G. S. Sukhatme, "Self-calibrating multi-sensor fusion with probabilistic measurement validation for seamless sensor switching on a UAV," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 4289–4296, 2016.

[6] C. Tessier, C. Cariou, C. Debain, F. Chausse, R. Chapuis, and C. Rousset, "A real-time, multi-sensor architecture for fusion of delayed observations: application to vehicle localization," in *2006 IEEE Intelligent Transportation Systems Conference*. IEEE, 2006. [Online]. Available: https://doi.org/10.1109/itsc.2006.1707405

[7] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, Jul. 2014.

[8] M. Darms and H. Winner, "A modular system architecture for sensor data processing of ADAS applications," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2005, pp. 729–734, 2005.

[9] D. A. Cucci and M. Matteucci, "On the Development of a Generic Multi-Sensor Fusion Framework for Robust Odometry Estimation," *Journal of Software Engineering for Robotics*, vol. 5, no. May, pp. 48–62, 2014.

[10] 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," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2014.

[11] S. I. Roumeliotis and G. A. Bekey, "Distributed multirobot localization," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 781–795, 2002.

[12] P. J. Rousseeuw and G. Molenberghs, "Transformation of non positive semidefinite correlation matrices," *Communications in Statistics - Theory and Methods*, vol. 22, no. 4, pp. 965–984, Jan. 1993. [Online]. Available: https://doi.org/10.1080/03610928308831068

[13] R. Rebonato and P. Jaeckel, "The most general methodology to create a valid correlation matrix for risk management and option pricing purposes," *SSRN Electronic Journal*, 2011. [Online]. Available: https://doi.org/10.2139/ssrn.1969689

[14] N. J. Higham, "Computing a nearest symmetric positive semidefinite matrix," *Linear Algebra and its Applications*, vol. 103, pp. 103–118, May 1988. [Online]. Available: https://doi.org/10.1016/0024-3795(88)90223-6

[15] X. R. Li, Z. Zhao, and X. B. Li, "Evaluation of Estimation Algorithms: Credibility Tests," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 42, no. 1, pp. 147–163, 2012.

[16] W. J. Hughes, "Global positioning system (gps) standard positioning service (sps) performance analysis report," *Tech. Cntr. NSTB/WAAS T and E Team*, no. 87, 2014.

[17] L. Serrano, D. Kim, R. B. Langley, K. Itani, and M. Ueno, "A gps velocity sensor: how accurate can it be?–a first look," in *ION NTM*, vol. 2004, 2004, pp. 875–885.