Fall 2018

Multitarget tracking and terrain-aided navigation using square-root consider filters

James Samuel McCabe

Department: Mechanical and Aerospace Engineering

MULTITARGET TRACKING AND TERRAIN-AIDED NAVIGATION USING SQUARE-ROOT CONSIDER FILTERS

by

JAMES SAMUEL MCCABE

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

AEROSPACE ENGINEERING

2018

Approved by

Kyle J. DeMars, Advisor
Henry Pernicka
Serhat Hosder
Robert Paige
Christopher D'Souza

ABSTRACT

Filtering is a term used to describe methods that estimate the values of partially observed states, such as the position, velocity, and attitude of a vehicle, using current observations that are corrupted due to various sources, such as measurement noise, transmission dropouts, and spurious information. The study of filtering has been an active focus of research for decades, and the resulting filters have been the cornerstone of many of humankind's greatest technological achievements. However, these achievements are enabled principally by the use of specialized techniques that seek to, in some way, combat the negative impacts that processor roundoff and truncation error have on filtering.

Two of these specialized techniques are known as square-root filters and consider filters. The former alleviates the fragility induced from estimating error covariance matrices by, instead, managing a factorized representation of that matrix, known as a square-root factor. The latter chooses to account for the statistical impacts a troublesome system parameter has on the overall state estimate without directly estimating it, and the result is a substantial reduction in numerical sensitivity to errors in that parameter. While both of these techniques have found widespread use in practical application, they have never been unified in a common square-root consider framework. Furthermore, consider filters are historically rooted to standard, vector-valued estimation techniques, and they have yet to be generalized to the emerging, set-valued estimation tools for multitarget tracking.

In this dissertation, formulae for the square-root consider filter are derived, and the result is extended to finite set statistics-based multitarget tracking tools. These results are used to propose a terrain-aided navigation concept wherein data regarding a vehicle's environment is used to improve its state estimate, and square-root consider techniques provide the numerical stability necessary for an onboard navigation application. The newly developed square-root consider techniques are shown to be much more stable than standard formulations, and the terrain-aided navigation concept is applied to a lunar landing scenario to illustrate its applicability to navigating in challenging environments.

ACKNOWLEDGMENTS

First things first, I would like to thank my advisor, Dr. Kyle DeMars, for enabling the research that fills these pages and without whom I would have been left with insufficient technical, computational, and financial resources to complete this research. I would also like to thank the members of my committee at Missouri S&T and NASA Johnson Spaceflight Center, namely Drs. Henry Pernicka, Serhat Hosder, Robert Paige, and Christopher D'Souza. Your influence both inside and outside of the classroom will always have a lasting impact on my own personal and professional development. Thank you all.

I would like to sincerely thank the NASA Space Technology Research Fellowship for funding my tenure as a graduate student at Missouri S&T.

I would like to list the names of everyone who has positively impacted my life over the years, but I am fortunate enough to be blessed with so many wonderful friends and influences that I could fill a dozen pages with their names. Instead, I'll thank those of you who studied those countless hours with me, endured my penchant for technical rambling, produced—and groaned with me at—the most dreadful puns, and served as a sounding board for new ideas. This is a heartfelt thanks to everyone who laughed with me in the hardest times, when laughing was the only recourse, and who never allowed me to forget who I really am. You know who you are.

I would like to thank my family for their ceaseless support of my goals and aspirations, no matter where my compass pointed. Any time I needed help, be it financial or spiritual, you were there. I am certain that none of this would have been possible without our constant laughter around the dinner table, immensely patriotic boating excursions, and hundred degree summers, sweating with a paintbrush in my hand and learning the true meaning of hard work. My parents, my sister Madisen, and the rest of my family are principally responsible for the man I am today, so if this dissertation contains anything of value, they deserve the credit. Finally, a dedication:

To my parents, Michael and Stephanie Coleman.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

LIST OF ALGORITHMS

1. INTRODUCTION

They never did understand that we were not going to rely on ground-based measurements, that we were going to make those measurements on board the spacecraft. I don't need to know the latitude and longitude of New York City to get there. I can just drive there, as long as I can see where I'm going.

– Richard H. Battin

1.1. MOTIVATION

Human curiosity engenders an intrinsic predisposition for exploration and discovery. As monumental advances in science and technology transfer from concept to point of fact, increasingly ambitious goals take their place that pose new, progressively more challenging problems. One of an engineer's principal tasks is to tread the fine line between theory and practice to successfully, and reliably, marry theoretical abstraction with real-world implementation. One such problem is that of filtering, and this topic serves as the focus of this dissertation.

Modeling the motion of some objects in the universe, be they natural entities, such as celestial bodies, or things of human devising, such as automobiles, aircraft, or spacecraft, has been a focus of research for centuries. The motion of these objects is nearly always stochastic and uncertain, prohibiting a deterministic conclusion upon where that object is, or is not, in space, even if perfect knowledge of that object was available in the past. Filtering refers to inferring estimates of an object's state, such as position, velocity, and/or attitude, given imperfect, partial observations of that object as it evolves through time. Some problems are tasked with a non-trivial extension to estimating the states of an unknown, and time-varying, number of objects simultaneously. Regardless of the problem at hand, this filtering

process must be capable of processing data collected on the object(s) that is inherently corrupted with noise, biases, and detection artifacts to, statistically and/or probabilistically, obtain an improved understanding of the object(s) of interest.

Many, if not most, applications of filtering theory to real-world problems critically depend upon reliable operation of the filter such that an accurate state estimate is available when it is necessary. For example, if the minds responsible for the successful recovery of the Apollo 13 capsule were denied estimates of where the capsule was in space, they would have literally been "flying blind," most likely resulting in complete loss of the vehicle and astronauts. Failure of a given system to reliably provide an accurate state estimate at any given time compromises that system's decision-making authority and, therefore, inherently represents a critical failure of that system.

Two key elements are identified in the preceding paragraph: accurate and reliable. Accuracy of a state estimate refers to not only providing an estimate as to the object's state, but also, somewhat more importantly, quantifying the uncertainty in that estimate. Usually, this uncertainty is quantified using covariance matrices, and, from a design perspective, the analytical accuracy of a covariance estimate is usually guaranteed by careful design of the employed filter models, i.e. carefully modeling the dynamical and observational processes of the problem. Reliability, on the other hand, is where theory and practice tend to part ways, and it turns out that filtering applications are subject to devastating failure due to digital computing errors in the required covariance computations. While filters are theoretically unhindered by the use of covariance matrices as an uncertainty representation, numerical errors induced by roundoff and truncation within a processor lead to catastrophic failure of many filter implementations.

A substantial portion of the research dedicated to addressing this lack of numerical stability was conducted under the context of navigation for spaceflight. Aerospace applications are largely responsible for the now-universal use of filters, most famously the Kalman filter [1, 2], and it is likely that they are responsible for many of the modern developments in filtering as well. These applications are the core inspiration for two now widely-used techniques that improve the numerical stability of the Kalman filter, known as square-

root filtering and consider filtering. These techniques are described in detail later in this dissertation, but their key feature is that they attempt to directly address the numerical phenomenon that tend to cause filter failure.

Square-root filtering and consider filtering have enabled successful applications in the past, but as mission designs become increasingly ambitious, navigation challenges correspondingly inflate with that ambition. Continued lunar exploration, space stations orbiting the Moon, and manned Mars occupation are all seemingly on the horizon for human exploration, and as human beings delve deeper into the inhospitable and unforgiving reaches of space, new challenges present themselves that demand enhanced filtering and navigation performance. In particular, future spaceflight beyond low Earth orbit necessitates navigation capabilities in environments where critical data types that are commonly relied upon, such as communication with the Global Positioning System or frequent Earth-based observations with distributed radar dishes, are entirely unavailable.

New paradigms must be explored, and this dissertation investigates the use of novel multitarget filtering strategies to enable terrain-aided navigation in challenging and complex environments. This technique utilizes features within a vehicle's surroundings, such as rocks, craters, beacons, or other spacecraft, as sources of information to reduce state uncertainty. To enable numerically stable application, square-root filtering and consider filtering are married under a common architecture for the first time, and the result is generalized to the multitarget domain to accommodate the terrain aiding concept. In other words, this dissertation contributes novel formulation and analysis of new filtering architectures for single-target tracking, multitarget tracking, and terrain-aided navigation using square-root consider filters.

1.2. CONTRIBUTIONS

The key original contributions of this dissertation are:

- The new derivation of the square-root consider filter,
- mathematical formalisms that generalize consider filtering to the Bayesian and multitarget tracking domains,

- a proposed strategy for practical terrain-aided navigation with finite set statistics,
- a specialized technique for augmenting classical techniques with the new methods,
- the successful numerical implementation and verification of all of the new results, and
- the first analysis of the aforementioned technologies for spacecraft navigation design, namely planetary descent and landing.

1.3. PUBLICATIONS

The research described herein has produced the following publications:

[1] James S. McCabe and Kyle J. DeMars. Terrain-aided navigation with decentralized fusion and finite set statistics. *NAVIGATION, Journal of the Institute of Navigation*, 2018. (Submitted).

[2] James S. McCabe and Kyle J. DeMars. Multiple set filtering using probability hypothesis densities. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, 2018.

[3] James S. McCabe and Kyle J. DeMars. Fusion methodologies for orbit determination with distributed sensor networks. In *21st International Conference on Information Fusion (FUSION)*. IEEE, 2018.

[4] James S. McCabe and Kyle J. DeMars. Square-root consider filters with hyperbolic Householder reflections. *Journal of Guidance, Control, and Dynamics*, 41(10):2098–2111, 2018. doi: 10.2514/1.G003417.

[5] James S. McCabe and Kyle J. DeMars. Robust, terrain-aided landing navigation through decentralized fusion and random finite sets. In *2018 AIAA Guidance, Navigation, and Control Conference*, 2018. doi: 10.2514/6.2018-1332.

[6] James S. McCabe and Kyle J. DeMars. Considering uncertain parameters in non-Gaussian estimation for single-target and multitarget tracking. *Journal of Guidance, Control, and Dynamics*, 40(9):2138–2150, 2017. doi: 10.2514/1.G002785.

[7] James S. McCabe and Kyle J. DeMars. Decentralized fusion with finite set statistics for landing navigation. In *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference*, 2017.

[8] James S. McCabe and Kyle J. DeMars. Feature-based robotic mapping with generalized labeled multi-Bernoulli filters for planetary landers. In *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference*, 2016. doi: 10.2514/6.2016-5565.

[9] James S. McCabe, Kyle J. DeMars, and Carolin Früh. Integrated detection and tracking for multiple space objects. In *Proceedings of the AAS/AIAA 24th Space Flight Mechanics Meeting*, Advances in the Astronautical Sciences, 2015.

1.4. A HISTORICAL PERSPECTIVE

In 1960, the world would be forever changed as Rudolf E. Kalman published his now infamous solution to the state-space linear filtering and prediction problem [1]. At the time, Kalman was working on United States Air Force sponsored research at the Research Institute for Advanced Studies in Baltimore, Maryland, and, with the help of Richard S. Bucy, managed to re-imagine two decades-old optimal estimation strategies from the frequency domain into the time domain [2]. The result, now universally known as the "Kalman filter" has had a tremendous impact on the development of technology and is at the core of some of the most tremendous human achievements, such as multiple manned expeditions to the Moon, the Global Positioning System, reusable manned vehicles such as the Space Shuttle, constant occupation of low Earth orbit via the International Space Station, an unprecedented precision in vehicle manufacturing, an impressive delivery cadence of payloads to orbit with launch vehicles, and burgeoning capabilities in vehicle autonomy. These are solely examples within spaceflight, but the Kalman filter has propagated into nearly every conceivable arena, from multi-million dollar medical imaging equipment down to the cell phone in nearly everyone's pocket.

Dr. Kalman deserves every bit of praise and recognition received for his tremendous impact on technology,–eventually receiving the Charles Stark Draper Prize and the National Medal of Science, among others–but the filter's discovery actually preceded his contribution. In 1880, astronomer Thorvald N. Thiele published the filter for solving geodesy problems by sequentializing least squares [12], but he and his peers were unable understand its immense value, as on-line computing was inconceivable at the time. This discovery would go largely unnoticed for nearly 80 years, somewhat due to it being overshadowed by the remainder of Thiele's own work, when Peter Swerling proposed effectively the same filter for tracking satellites in 1958 [13]. Richard Battin proposed its use, albeit in a less general context, in a 1960 technical report [14] without ever being aware of Kalman or his work, and he may be responsible for first "decoding" Kalman's advanced theory and mathematics for early practicing engineers [15, 16]. In fact, most modern derivations of the Kalman filter more closely follow Battin's direct variance minimization rather than Kalman's orthogonal

projections. This is all to say that there seems to be a somewhat inexplicable "gravitational pull" toward the equations defining the Kalman filter. Fortunately, Kalman and Battin in particular were just in time, because soon thereafter, John F. Kennedy announced that the United States would "commit itself to achieving the goal, before this decade is out, of landing a man on the Moon and returning him safely to the Earth." He went on to say that "No single space project in this period will be more impressive to mankind, or more important in the long-range exploration of space; and none will be so difficult or expensive to accomplish."

Truer words have rarely been spoken, and approximately $400,000$ engineers, technicians, and scientists got to work solving the seemingly insurmountable challenges posed by the Apollo program. James Webb reportedly called Charles Draper, founder and director of the MIT Instrumentation Laboratory, offering the very first Apollo contract to develop a guidance and navigation system and saying that he thought it was "one of our hardest problems. Do you think that you could help us with that?" [17]. Draper responded "Of course." and the resulting contract would not only serve as a major component enabling the success of the Apollo program but serve as the catalyst that took the Kalman filter from a theoretical abstraction to a practical methodology for filtering and onboard navigation.

A key player in the development and dissemination of the Kalman filter was Stanley Schmidt of NASA Ames who was tasked with developing a real-time data processor for vehicle state estimation at the Moon. In 1960, Kalman invited himself to Ames and Schmidt, who had known Kalman for several years, found Kalman's linear filtering theory worth investigating [16]. Schmidt immediately set his team to task on interrogating the features of the filter, and, within only a few years, it is these interrogations that would ultimately produce the game-changing techniques that enable reliable, real-time implementation of Kalman filters, such as developing the ubiquitous extended Kalman filter, decomposition of the Kalman filter into separate time and measurement update stages, numerous discoveries on clever and efficient computer implementations, and consider (or Schmidt-Kalman) filtering [16, 18]. Beyond that, Schmidt and his staff would serve as the primary vehicle

that equipped Battin and the MIT Instrumentation Laboratory with the knowledge and capabilities of the Kalman filter, ultimately leading to its integration in to the Apollo flight computer.

A much lesser known, yet immensely influential, early adopter and innovator of the Kalman filter is William "Bill" Lear, and his lack of acclaim can largely be attributed to the fact that his body of work, while enormous, is largely contained in unpublished memos and monographs. In the 1960s, Lear was with the TRW Systems Group working on an Apollo contract to develop a ground-based navigation system dubbed the Powered Flight Processor [19]. This system was almost entirely designed by Lear himself, even providing complete Fortran code to the IBM computer scientists tasked with implementing it [20]. The effect Lear's acuity had on his peers was exemplified during Apollo 11's descent toward the lunar surface, when NASA ground controllers found that the solutions provided by Lear's filter and their internally-developed data processor began to disagree by thousands of feet, and they accepted Lear's as the operational solution [20]. This filter began to become known locally as the "Lear filter" and was used for every subsequent Apollo expedition. Lear's dissertation, published in 1965 at Purdue University, is concerned with tracking orbiting objects and is one of the earliest, and most insightful, applications of the Kalman filter to extremely challenging nonlinear systems [21]. His dissertation contains quite possibly the earliest reference to and description of a technique now known as residual editing, as well as being among the earliest uses of exponentially correlated random variables to model uncertain parameters. Additionally, his memo describing a proposed multi-phase navigation scheme for the Space Shuttle appears to be the earliest reference to a technique now known as underweighting [22]. Each of these techniques are widely applied to most real-time filtering and navigation applications today.

It is unfair to stop here, because numerous others have contributed to the key developments that enable the widespread success and celebration of the Kalman filter, and the preceding discussion is a gross underrepresentation of the rich tapestry of innovation that led to this point. However, the point is that as long as Kalman filters have been applied, the difference between theory and practice has led to catastrophic failure in implementation, and substantial effort has been, and continues to be, applied to hardening filters to numer-

ical degradation. A principal focus of this dissertation is in creating new techniques that enable stable filtering and navigation applications, and these developments are only possible due to the vantage point afforded by standing upon the shoulders of giants. Throughout this document, a concerted effort has been expended to provide a reader with appropriate historical context and to spotlight the tremendous contributions of some lesser-known innovators in this field.

1.5. ORGANIZATION OF THIS DISSERTATION

This dissertation is organized as follows:

- Section 2 provides a description of traditional minimum mean square error estimation, a selection of practical navigation techniques, and square-root filters.
- Section 3 describes new advances in consider filtering contributed by this work. In particular, the square-root consider filter is derived, and an analog to MMSE consider filtering is derived using Bayes' rule to enable non-Gaussian Bayesian consider filters.
- Section 4 extends consider filtering to the realm of multitarget tracking, and the consider probability hypothesis density (PHD) filter is derived. The section continues by proposing a terrain-aided navigation concept using multitarget tracking, and a scheme for augmenting traditional navigation schemes with the newly proposed techniques using fusion is presented. The section concludes by deriving a new filter using PHDs for simultaneous estimation of multiple sets, and the result is applied to the proposed terrain-aided navigation concept.
- Section 5 explores the use of the techniques developed in the previous sections for planetary landing navigation. In particular, salient features of the lunar landing problem are discussed, and simulations are presented to evaluate the performance of the proposed techniques.
- Section 6 presents concluding remarks and discusses future research directions.

A savvy practitioner will notice that this dissertation, while largely focused on navigation, hardly contains a comprehensive discussion regarding some of the elements necessary for most real-world, practical navigation, such as inertial measurement unit-based time updates, coning, sculling, attitude estimation, measurement delays, sensor modeling, flight processor considerations, and what have you. Describing every useful tool in the navigator's toolbox would be a Sisyphean task that this work makes no effort to accomplish. Instead, this dissertation intends to lean upon the vast literature on the aforementioned topics, rather than recounting their details, and aims to limit discussion to important topics that are either absent from literature or, in the author's opinion, in need of clarity and emphasis.

Comments on Notation and Terminology. This dissertation attempts to adopt notation and terminology as transparent and consistent as possible such that a reader can move from section to section, and section to section, with minimal risk of confusion, and such that a reader can directly, or nearly directly, substitute the results of, say, Section 3.2 into Section 4.3. Scalar quantities are typically written using non-bold, lowercase Roman and Greek characters, such as a or γ, and vector quantities are typeset as bold, lowercase Roman and Greek characters, such as $\boldsymbol{a}$ or $\boldsymbol{\gamma}$. Matrix quantities are denoted as bold, uppercase Roman and Greek characters, such as $\boldsymbol{A}$ or $\boldsymbol{\Gamma}$, and later portions of this dissertation will represent set-valued terms in the same way where there is no risk of confusion. Otherwise, standard mathematical conventions are sought to be adhered to, such as $\mathbb{E}\{\cdot\}$ denoting the expected value operation, $\mathbb{R}^n$ denoting the standard real coordinate space in n dimensions, the expression $a \leftarrow a'$ denoting "a is overwritten with a'", etc.

For some vector $\boldsymbol{x}_k$, its mean at k^{th} time index is denoted $\boldsymbol{m}_{x,k}$, and its associated error covariance is denoted by $\boldsymbol{P}_{xx,k}$. If a quantity is adorned with a "$-$" or a "$+$", such as $\boldsymbol{m}_{x,k}^{-}$ and $\boldsymbol{m}_{x,k}^{+}$, it is designated an *a priori* quantity (i.e. before a measurement is processed) and an *a posteriori* quantity (i.e. after a measurement is processed), respectively. The square-root factor of $\boldsymbol{P}_{xx,k}$ is denoted $\boldsymbol{S}_{xx,k}$, and this will be discussed at length later. If another vector $\boldsymbol{c}_k$ is taken into consideration, it is treated the same way, and the correlations matrix between $\boldsymbol{x}_k$ an $\boldsymbol{c}_k$ is denoted $\boldsymbol{P}_{xc,k}$. Some typical descriptions of process models take

the form

$$\boldsymbol{x}_k = \boldsymbol{F}_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{L}_{k-1}\boldsymbol{w}_{k-1}\,,$$

but, instead, this written here as

$$\boldsymbol{x}_k = \boldsymbol{F}_{x,k-1}\boldsymbol{x}_{k-1} + \boldsymbol{F}_{w,k-1}\boldsymbol{w}_{k-1}\,.$$

The reason for this difference is that the former method can produce an "alphabet soup" of terms, where a variety of different characters denoting transition matrices, noise mapping terms, state covariances, noise covariances, etc. appear, and a reader is forced to remember what matrix "$\boldsymbol{L}_{k-1}$" is supposed to denote. Instead, under this construction, the primary symbol conveys the general piece of information (i.e. any matrix $\boldsymbol{F}$ belongs to the transition dynamics of $\boldsymbol{x}_k$) and the subscript completes the information (i.e. $\boldsymbol{F}_{a,k-1}$ is the transition matrix corresponding to vector $\boldsymbol{a}_{k-1}$ at $k-1$). The trade-off, of course, is that readers accustomed to the typical descriptions might be confused at first, and the extra ornamentation complicates notation somewhat, but the author feels it is the setting that produces the most cohesive, consistent exposition requiring the least memorization.

For very good reason, it is conventional to express a filter in terms of two stages: a predictor stage and a corrector stage. The predictor stage governs the temporal motion of the system according to system dynamics, noise, and other forcing functions. The corrector stage is the mechanism for processing data generated about the state according to some corrupted process that is collected to improve state knowledge. Together, a predictor/corrector recursion provides a tool to repeatedly process data as time passes, and, therefore, (usually) completely defines a given filter.

Finally, as a point of clarity, it is noted that this dissertation will refer to minimum mean square error (MMSE) estimation and Bayesian estimation as belonging to different paradigms. The term "MMSE estimation" is used to denote methods that seek to directly estimate the error statistics, i.e. mean and covariance, of the state vector of interest without making any prescriptions upon the form of the underlying state density. The term

"Bayesian estimation" is used to denote those methods that, as a point of modeling, explicitly assume a form of the state and noise distributions, such as modeling them using Gaussian probability densities, and subsequently produce relationships for the parameters of those distributions. While, in general, (linear) MMSE estimation can be interpreted of as a subset of (nonlinear) Bayesian estimation, the distinction is made here for convenience of exposition and comparison to distinguish their very different derivations. For example, it is well known that one can obtain the very same Kalman filter equations using both of these approaches, and, though the result is the same, their assumptions, procedures, and interpretation are very different and, therefore, are distinguished.

2. TRADITIONAL APPROACHES TO FILTERING

This section describes traditional approaches to filtering and navigation, and it offers important conceptual, historical, and theoretical context to the later developments of this dissertation. Section 2.1 presents the predictor/corrector relationships derived under MMSE principles and specifically details how to apply these equations to real-world nonlinear systems using both linearization and quadrature-based approximations. Section 2.2 explores various practical techniques employed by navigators for numerically hardening a filtering implementation against the errors induced by finite precision computing. Section 2.3 elaborates on this point by presenting square-root forms of the MMSE filters derived in previous sections.

2.1. MINIMUM MEAN SQUARE ERROR ESTIMATION

Consider the nonlinear dynamical motion of an n_x-dimensional random vector $\boldsymbol{x}_k$, governed by

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1}) \,, \tag{2.1}$$

where $\boldsymbol{x}_k$ denotes the state at time t_k, $\boldsymbol{f}(\cdot,\cdot)$ is a nonlinear, discrete-time function defined by the equations of motion, and $\boldsymbol{w}_{k-1}$ belongs to a zero-mean, white process noise sequence that serves as stochastic excitation in the state. As the state evolves, partial observations of $\boldsymbol{x}_k$ are collected, denoted by $\boldsymbol{z}_k$, and are generated according to the model

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k) \,, \tag{2.2}$$

where $\boldsymbol{h}(\cdot,\cdot)$ is the nonlinear measurement function and $\boldsymbol{v}_k$ belongs to a zero-mean, white measurement noise sequence that corrupts the collected data. Each measurement is of dimension n_z, and the dimensions of the process and measurement noise vectors are denoted as n_w and n_v, respectively. It is assumed that the covariances of these zero-mean, white

noise sequences are exactly known, denoted as

$$\boldsymbol{P}_{ww,k-1} = \mathbb{E}\left\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\right\}$$
$$\boldsymbol{P}_{vv,k} = \mathbb{E}\left\{\boldsymbol{v}_k\boldsymbol{v}_k^T\right\} ,$$

where $\mathbb{E}\{\cdot\}$ denotes the mathematical expectation and $(\cdot)^T$ denotes the transpose. In addition, it is assumed that these noises are uncorrelated with each other as well as the state itself. The initial mean and covariance of the state are assumed known and are given by

$$\boldsymbol{m}_{x,0} = \mathbb{E}\{\boldsymbol{x}_0\} \tag{2.3a}$$
$$\boldsymbol{P}_{xx,0} = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})^T\right\} . \tag{2.3b}$$

In what follows, the superscript "$-$" is be used to denote an *a priori* quantity (i.e. prior, immediately before a measurement is processed) and the superscript "$+$" is used to denote an *a posteriori* quantity (i.e. posterior, immediately after a measurement is processed).

The MMSE filter derived here (the Kalman filter) is that which obtains the posterior mean estimate as a linear combination of the measurements and the state. In fact, it is the principles of linear MMSE estimation that are discussed here, and this should not be confused as being a linearity constraint upon Eqs. (2.1) and (2.2). Indeed, the following methods can be adapted for application in nonlinear systems, and two popular methods for this will be discussed in Sections 2.1.4 and 2.1.5 (linearization and quadrature, respectively). This filter is linear in that it linearly combines data and *a priori* information, and careful selection of minimization criteria produces the celebrated Kalman filter.

2.1.1. Predictor. The predicted mean of the state is computed by taking the expected value of Eq. (2.1) as

$$\boldsymbol{m}_{x,k}^- = \mathbb{E}\{\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1})\} , \tag{2.4}$$

and the state error covariance is found by taking the expected value of the mean-square deviations from the mean, yielding

$$\boldsymbol{P}_{xx,k}^{-} = \mathbb{E}\left\{(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}_{x,k}^{-})(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}_{x,k}^{-})^T\right\} . \tag{2.5}$$

2.1.2. Corrector. With the *a priori* quantities obtained, it remains to incorporate new measurement data, $\boldsymbol{z}_k$, to produce the *a posteriori* state estimates. As alluded to previously, the posterior state is taken to be some linear combination of the measurement data, written generally as

$$\boldsymbol{m}_{x,k}^{+} = \boldsymbol{a}_k + \boldsymbol{K}_{x,k}\boldsymbol{z}_k \,,$$

where $\boldsymbol{a}_k$ and $\boldsymbol{K}_{x,k}$ are parameters that must be determined. If one defines the posterior estimation error as

$$\boldsymbol{e}_k^{+} = \boldsymbol{x}_k - \boldsymbol{m}_{x,k}^{+} \,,$$

and takes the posterior solution to be unbiased, i.e. $\mathbb{E}\left\{\boldsymbol{e}_k^{+}\right\} = \boldsymbol{0}_{n_x \times 1}$, the resulting linear, unbiased estimator is given as

$$\boldsymbol{m}_{x,k}^{+} = \boldsymbol{m}_{x,k}^{-} + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^{-}) \,, \tag{2.6}$$

where the term $\boldsymbol{m}_{z,k}$ is the expected value of the measurement function,

$$\boldsymbol{m}_{z,k}^{-} = \mathbb{E}\left\{\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k)\right\} . \tag{2.7}$$

What remains, then, is to determine the parameter $\boldsymbol{K}_{x,k}$ such that, true to the filter's designation, the posterior minimum mean-square state estimation error is minimized. Using the posterior state estimation error, and the definition of the posterior error covariance

$$\boldsymbol{P}_{xx,k}^{+} = \mathbb{E}\left\{\boldsymbol{e}_k^{+}(\boldsymbol{e}_k^{+})^T\right\} ,$$

it can be shown that the posterior error covariance can be written as [23]

$$\boldsymbol{P}^{+}_{xx,k} = \boldsymbol{P}^{-}_{xx,k} - \boldsymbol{P}^{-}_{xz,k}\boldsymbol{K}^{T}_{x,k} - \boldsymbol{K}_{x,k}(\boldsymbol{P}^{-}_{xz,k})^{T} + \boldsymbol{K}_{x,k}\boldsymbol{P}^{-}_{zz,k}\boldsymbol{K}^{T}_{x,k}\,, \tag{2.8}$$

where

$$\boldsymbol{P}^{-}_{xz,k} = \mathbb{E}\left\{(\boldsymbol{x}_k - \boldsymbol{m}^{-}_{x,k})(\boldsymbol{z}_k - \boldsymbol{m}^{-}_{z,k})^{T}\right\} \tag{2.9a}$$

$$\boldsymbol{P}^{-}_{zz,k} = \mathbb{E}\left\{(\boldsymbol{z}_k - \boldsymbol{m}^{-}_{z,k})(\boldsymbol{z}_k - \boldsymbol{m}^{-}_{z,k})^{T}\right\}\,. \tag{2.9b}$$

The matrices $\boldsymbol{P}_{xz,k}$ and $\boldsymbol{P}_{zz,k}$ are called the cross covariance and the residual covariance, respectively, and can hold huge roles in a practical implementation (such as the use of the residual covariance discussed later in Section 2.2.1). The gain $\boldsymbol{K}_{x,k}$ is found such that the objective

$$J = \mathbb{E}\left\{(\boldsymbol{e}^{+}_{k})^{T}\boldsymbol{e}^{+}_{k}\right\} = \mathrm{tr}\{\boldsymbol{P}^{+}_{xx,k}\}$$

is minimized, where $\mathrm{tr}\{\cdot\}$ denotes the matrix trace. This is where the principles of MMSE-based estimation are applied, and solving for $\boldsymbol{K}_{x,k}$ produces the well known solution

$$\boldsymbol{K}_{x,k} = \boldsymbol{P}^{-}_{xz,k}(\boldsymbol{P}^{-}_{zz,k})^{-1}\,, \tag{2.10}$$

a parameter most commonly referred to as the Kalman gain.

Summary. The general formulation of the MMSE filter is shown in Table 2.1. Given the initial mean and covariance in Eqs. (2.3), the *a priori* mean and covariance are obtained with Eqs. (2.4) and (2.5). Then, given the measurement $\boldsymbol{z}_k$, the *a posteriori* mean and covariance are found using Eqs. (2.6) and (2.8). This recursion fully defines the filter, and all that remains for a user is to evaluate the required expectations. In the following sections, methods for solving these expectations for arbitrary nonlinear systems are briefly outlined.

Table 2.1. General formulation of the MMSE filter.

Models	$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1})$	$\boldsymbol{P}_{ww,k-1} = \mathbb{E}\left\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\right\}$
	$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k)$	$\boldsymbol{P}_{vv,k} = \mathbb{E}\left\{\boldsymbol{v}_k\boldsymbol{v}_k^T\right\}$
Initialization	$\boldsymbol{m}_{x,0} = \mathbb{E}\left\{\boldsymbol{x}_0\right\}$	$\boldsymbol{P}_{xx,0} = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})^T\right\}$
Predictor	$\boldsymbol{m}_{x,k}^- = \mathbb{E}\left\{\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1})\right\}$	
	$\boldsymbol{P}_{xx,k}^- = \mathbb{E}\left\{(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}_{x,k}^-)(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}_{x,k}^-)^T\right\}$	
Corrector	$\boldsymbol{m}_{x,k}^+ = \boldsymbol{m}_{x,k}^- + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^-)$	
	$\boldsymbol{P}_{xx,k}^+ = \boldsymbol{P}_{xx,k}^- - \boldsymbol{P}_{xz,k}^-\boldsymbol{K}_{x,k}^T - \boldsymbol{K}_{x,k}(\boldsymbol{P}_{xz,k}^-)^T + \boldsymbol{K}_{x,k}\boldsymbol{P}_{zz,k}^-\boldsymbol{K}_{x,k}^T$	
	$\boldsymbol{K}_{x,k} = \boldsymbol{P}_{xz,k}^-(\boldsymbol{P}_{zz,k}^-)^{-1}$	
	$\boldsymbol{m}_{z,k}^- = \mathbb{E}\left\{\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k)\right\}$	
	$\boldsymbol{P}_{zz,k}^- = \mathbb{E}\left\{(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^-)(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^-)^T\right\}$	
	$\boldsymbol{P}_{xz,k}^- = \mathbb{E}\left\{(\boldsymbol{x}_k - \boldsymbol{m}_{x,k}^-)(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^-)^T\right\}$	

2.1.3. Various Forms of the Covariance Update. While the covariance update of Eq. (2.8) is a valid, and quite general, form of the error covariance update, research since the Kalman filter's inception in the 1960's has produced a number of variants. Each form is, in some sense, equivalent, and can possess different analytical and numerical implications. Their differences are quite nuanced, often being qualitative claims on numerical characteristics. Others are crucial to theoretical interpretation, such as noting the fact that Eq. (2.8) is true for *any* gain $\boldsymbol{K}_{x,k}$ whereas the to-appear Eq. (2.11) is only true for the *Kalman* gain of Eq. (2.10).

Among the earliest, and generally the most popularly published forms of the covariance update is for the case of linear systems, i.e. measurement schemes of the form

$$\boldsymbol{z}_k = \boldsymbol{H}_{x,k}\boldsymbol{x}_k + \boldsymbol{H}_{v,k}\boldsymbol{v}_k \,,$$

and is given by

$$\boldsymbol{P}_{xx,k}^+ = (\boldsymbol{I}_{n_x \times n_x} - \boldsymbol{K}_{x,k}\boldsymbol{H}_{x,k})\boldsymbol{P}_{xx,k}^- \,. \tag{2.11}$$

Despite being so widespread, this form of the covariance update faces many issues in numerical implementation (primarily the loss of symmetry and/or positive-semidefiniteness in $\boldsymbol{P}^{+}_{xx,k}$, both of which are required properties of a proper covariance matrix), and this topic is a large point of conversation of this dissertation (Section 2.3 and Section 3, in particular).

When practitioners and navigators began implementing Kalman filters on mainframe computers and in real-time applications, such as for the Apollo and C-5 programs, limited computational resources demanded maximum efficiency from the software being developed. By substituting Eq. (2.10) into Eq. (2.11) and right-distributing $\boldsymbol{P}^{-}_{xx,k}$, one is able to rewrite Eq. (2.11) for linear (or, indeed, *linearized*) systems as

$$\boldsymbol{P}^{+}_{xx,k} = \boldsymbol{P}^{-}_{xx,k} - \boldsymbol{P}^{-}_{xz,k}\boldsymbol{K}^{T}_{x,k}\,. \tag{2.12}$$

It turns out that this form of the covariance update is very inexpensive to compute and store, relatively speaking, though it shares the same numerical issues of Eq. (2.11). Early implementations utilized clever memory writing techniques to make this form as efficient as possible [19, 22, 24].

The numerical complications often plaguing filters that use Eq. (2.12) drove practitioners to investigate other techniques. Some practical modifications developed to combat these numerical complications (such as measurement underweighting discussed in Section 2.2.2 or the consider techniques of Section 3) often require non-Kalman, suboptimal gains such that Eqs. (2.11) and (2.12) are invalid. An expanded form of this covariance update that, in fact, is valid for any linear, unbiased estimator under standard modeling assumptions, is known as Joseph's form or Joseph's formula and is given as

$$\boldsymbol{P}^{+}_{xx,k} = (\boldsymbol{I}_{n_x\times n_x} - \boldsymbol{K}_{x,k}\boldsymbol{H}_{x,k})\boldsymbol{P}^{-}_{xx,k}(\boldsymbol{I}_{n_x\times n_x} - \boldsymbol{K}_{x,k}\boldsymbol{H}_{x,k})^{T} + \boldsymbol{K}_{x,k}\boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}^{T}_{v,k}\boldsymbol{K}^{T}_{x,k}\,. \tag{2.13}$$

This form of the covariance update takes its name from Peter D. Joseph who noticed that updates of the form of Eqs. (2.11) and (2.12) were sensitive to small errors in the computation of $\boldsymbol{K}_{x,k}$ (errors that, in some sense, are inevitable due to roundoff in the

required matrix inversion) [25].[1] Instead, Joseph advocated this expanded form that is accurate to roundoff errors to (at least) first order (see Section 16 of [25]).[2] It has come to widespread use within navigation applications despite its increased computational burden, and it has the advantage of being valid for any choice of update gain.

Another update found in literature is rather common in modern publication and given by

$$\boldsymbol{P}^{+}_{xx,k} = \boldsymbol{P}^{-}_{xx,k} - \boldsymbol{K}_{x,k}\boldsymbol{P}^{-}_{zz,k}\boldsymbol{K}^{T}_{x,k}\,. \tag{2.14}$$

This expression is true for arbitrary system models but requires the use of the optimal Kalman gain. This expression has gained popularity in recent years due to its usefulness within quadrature-based applications (such as in Section 2.1.5.1) and square-root filtering (see Section 2.3).

2.1.4. Linearization-based Approach. To apply the results of Sections 2.1.1 and 2.1.2 to real-world problems, one must evaluate the required expectations. To evaluate these expected values, the nonlinear dynamics and measurement model, Eqs. (2.1) and (2.2), respectively, can be expanded in a Taylor series and truncated above first order. The result is a linearization-based application of the MMSE filter known as the extended Kalman filter and is briefly described below.

2.1.4.1. Predictor. A first-order expansion of nonlinear dynamics about the posterior state estimate at t_{k-1} yields

$$\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1}) = \boldsymbol{f}(\boldsymbol{m}^{+}_{x,k-1}, \boldsymbol{0}_{n_w\times 1}) + \boldsymbol{F}_{x,k-1}(\boldsymbol{x}_{k-1} - \boldsymbol{m}^{+}_{x,k-1}) + \boldsymbol{F}_{w,k-1}\boldsymbol{w}_{k-1}\,,$$

[1] It is interesting to note that another early advocate of this form of the covariance update was Lear, as demonstrated in [21]. It is likely that Joseph and Lear were familiar, as they both were working for TRW Systems Group in Redondo Beach, with Lear later moving to work out of Houston at the NASA center locally. In his portion of [25], Joseph actually refers to Lear directly, suggesting that these two early innovators likely participated in conversation on the subject.

[2] In fact, Eq. (2.11) is a simplification of Eq. (2.13) for the optimal Kalman gain, serving as a cautionary tale: not all simplifications produce desirable results.

with Jacobian(s)

$$\boldsymbol{F}_{\alpha,k-1} = \left[\left. \frac{\partial \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1})}{\partial \boldsymbol{\alpha}_{k-1}} \right|_{(\cdot)^+_{k-1}} \right] .$$

It is to be understood that $\boldsymbol{\alpha}_{k-1}$ can represent $\boldsymbol{x}_{k-1}$ or $\boldsymbol{w}_{k-1}$, leading to the definitions of $\boldsymbol{F}_{x,k-1}$ and $\boldsymbol{F}_{w,k-1}$, respectively. The subscript $(\cdot)^+_{k-1}$ is used to indicate that each Jacobian is evaluated at the posterior means of the state, where a reader is asked to recall that $\boldsymbol{w}_{k-1}$ is taken to be zero-mean.

Applying this first-order Taylor series to the prediction equations of Eqs. (2.4) and (2.5) yields linearized expressions of the form

$$\boldsymbol{m}^-_{x,k} = \boldsymbol{f}(\boldsymbol{m}^+_{x,k-1}, \boldsymbol{0}_{n_w \times 1}) \tag{2.15a}$$

$$\boldsymbol{P}^-_{xx,k} = \boldsymbol{F}_{x,k-1} \boldsymbol{P}^+_{xx,k-1} \boldsymbol{F}^T_{x,k-1} + \boldsymbol{F}_{w,k-1} \boldsymbol{P}_{ww,k-1} \boldsymbol{F}^T_{w,k-1} \tag{2.15b}$$

where it is assumed that the posterior at t_{k-1} is unbiased.

2.1.4.2. Corrector. Much as with the dynamics, the nonlinear measurement function is expanded in a first-order Taylor series to yield

$$\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k) = \boldsymbol{h}(\boldsymbol{m}^-_{x,k}, \boldsymbol{0}_{n_v \times 1}) + \boldsymbol{H}_{x,k}(\boldsymbol{x}_k - \boldsymbol{m}^-_{x,k}) + \boldsymbol{H}_{v,k} \boldsymbol{v}_k \, ,$$

where

$$\boldsymbol{H}_{\gamma,k} = \left[\left. \frac{\partial \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k)}{\partial \boldsymbol{\gamma}_k} \right|_{(\cdot)^-_k} \right]$$

represents another general-form Jacobian. Here, $\boldsymbol{\gamma}_k$ can represent $\boldsymbol{x}_k$ and $\boldsymbol{v}_k$, leading to the definitions of $\boldsymbol{H}_{x,k}$ and $\boldsymbol{H}_{v,k}$, respectively, and the subscript $(\cdot)^-_k$ indicates that each Jacobian is evaluated at the prior means of the state, where it recalled that the measurement noise is taken to be zero mean. Applying this expansion to the expectations of Eq. (2.7)

Table 2.2. Linearization-based formulation of the MMSE filter.

Models	$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1})$	$\boldsymbol{P}_{ww,k-1} = \mathbb{E}\left\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\right\}$
	$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k)$	$\boldsymbol{P}_{vv,k} = \mathbb{E}\left\{\boldsymbol{v}_k\boldsymbol{v}_k^T\right\}$
Initialization	$\boldsymbol{m}_{x,0} = \mathbb{E}\left\{\boldsymbol{x}_0\right\}$	$\boldsymbol{P}_{xx,0} = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})^T\right\}$
Predictor	$\boldsymbol{m}_{x,k}^- = \boldsymbol{f}(\boldsymbol{m}_{x,k-1}^+, \boldsymbol{0}_{n_w\times 1})$	
	$\boldsymbol{P}_{xx,k}^- = \boldsymbol{F}_{x,k-1}\boldsymbol{P}_{xx,k-1}^+\boldsymbol{F}_{x,k-1}^T + \boldsymbol{F}_{w,k-1}\boldsymbol{P}_{ww,k-1}\boldsymbol{F}_{w,k-1}^T$	
Corrector	$\boldsymbol{m}_{x,k}^+ = \boldsymbol{m}_{x,k}^- + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^-)$	
	$\boldsymbol{P}_{xx,k}^+ = \boldsymbol{P}_{xx,k}^- - \boldsymbol{P}_{xz,k}^-\boldsymbol{K}_{x,k}^T - \boldsymbol{K}_{x,k}(\boldsymbol{P}_{xz,k}^-)^T + \boldsymbol{K}_{x,k}\boldsymbol{P}_{zz,k}^-\boldsymbol{K}_{x,k}^T$	
	$\boldsymbol{K}_{x,k} = \boldsymbol{P}_{xz,k}^-(\boldsymbol{P}_{zz,k}^-)^{-1}$	
	$\boldsymbol{m}_{z,k}^- = \boldsymbol{h}(\boldsymbol{m}_{x,k}^-, \boldsymbol{0}_{n_v\times 1})$	
	$\boldsymbol{P}_{zz,k}^- = \boldsymbol{H}_{x,k}\boldsymbol{P}_{xx,k}^-\boldsymbol{H}_{x,k}^T + \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}_{v,k}^T$	
	$\boldsymbol{P}_{xz,k}^- = \boldsymbol{P}_{xx,k}^-\boldsymbol{H}_{x,k}^T$	

and Eqs. (2.9) yields

$$\boldsymbol{m}_{z,k}^- = \boldsymbol{h}(\boldsymbol{m}_{x,k}^-, \boldsymbol{0}_{n_v\times 1}) \tag{2.16a}$$

$$\boldsymbol{P}_{xz,k}^- = \boldsymbol{P}_{xx,k}^-\boldsymbol{H}_{x,k}^T \tag{2.16b}$$

$$\boldsymbol{P}_{zz,k}^- = \boldsymbol{H}_{x,k}\boldsymbol{P}_{xx,k}^-\boldsymbol{H}_{x,k}^T + \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}_{v,k}^T\,, \tag{2.16c}$$

and evaluating the gain in Eq. (2.10) permits $\boldsymbol{m}_{x,k}^+$ and $\boldsymbol{P}_{xx,k}^+$ to be obtained with Eqs. (2.6) and (2.8).[3]

Summary. The linearization-based formulation of the MMSE filter is summarized in Table 2.2.

2.1.5. Quadrature-based Approach. Rather than approximating nonlinear functions with Taylor series, quadrature methods approximate expectations via some point-based integration scheme, Gauss-Hermite quadrature [26], cubature [27], or the unscented transform [28, 29]. Each rule has its own point and weight selection scheme, and so the following is treated as a common quadrature framework. An important detail of some methods, however, is that some permit negative weights for the points, and this has significant

[3]Instead of using Eq. (2.8), one could also refer to Section 2.1.3 for other expressions to obtain $\boldsymbol{P}_{xx,k}^+$.

implications on the positivity of the resulting error covariance matrix. This will be an important detail in Section 2.3.3, and developments in Section 3, particularly Section 3.2.4, offer significant improvements to these methods.

Given the nonlinear transformation

$$\boldsymbol{\alpha} = \boldsymbol{\varphi}(\boldsymbol{\beta}) \,,$$

where $\boldsymbol{\beta}$ has a known mean, $\boldsymbol{m}_\beta$, and covariance $\boldsymbol{P}_{\beta\beta}$, the mean and covariance of $\boldsymbol{\alpha}$, along with the cross-covariance between $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are approximated by

$$\boldsymbol{m}_\alpha = \sum_{\ell=1}^{q} w_m^{(\ell)} \boldsymbol{\varphi}(\boldsymbol{\beta}^{(\ell)}) \tag{2.17a}$$

$$\boldsymbol{P}_{\alpha\alpha} = \sum_{\ell=1}^{q} w_c^{(\ell)} (\boldsymbol{\varphi}(\boldsymbol{\beta}^{(\ell)}) - \boldsymbol{m}_\alpha)(\boldsymbol{\varphi}(\boldsymbol{\beta}^{(\ell)}) - \boldsymbol{m}_\alpha)^T \tag{2.17b}$$

$$\boldsymbol{P}_{\beta\alpha} = \sum_{\ell=1}^{q} w_c^{(\ell)} (\boldsymbol{\beta}^{(\ell)} - \boldsymbol{m}_\beta)(\boldsymbol{\varphi}(\boldsymbol{\beta}^{(\ell)}) - \boldsymbol{m}_\alpha)^T \,, \tag{2.17c}$$

where $\boldsymbol{\beta}^{(\ell)}$ are the quadrature points, $w_m^{(\ell)}$ and $w_c^{(\ell)}$ are the quadrature weights for the mean and covariance, respectively, and q is the number of quadrature points.

When the nonlinear function under consideration has multiple inputs, i.e. when it takes the form

$$\boldsymbol{\alpha} = \boldsymbol{\varphi}(\boldsymbol{\beta}, \boldsymbol{\gamma}) \,,$$

quadrature techniques are still applicable. In this case, the inputs are concatenated to form a single input $(\boldsymbol{\beta}')^T = [\boldsymbol{\beta}^T \;\; \boldsymbol{\gamma}^T]$. The mean and the covariance of the composite input are determined, including the cross-covariance between $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, and the previously described approach to determining the mean, covariance, and cross-covariance (with all inputs) of $\boldsymbol{\alpha}$ is utilized. When one, or more, of the inputs appear linearly, analytic integration can be, at least partially, used to transform the mean and covariance. This, however, is a specific case that is easily handled, so it will not be considered moving forward. If more than two inputs are required for a given transformation, the same approach of concatenation is applied.

2.1.5.1. Predictor. Implementing the prediction step requires that the expectations Eqs. (2.4) and (2.5) be evaluated. As the state dynamics, given by Eq. (2.1), contain a stochastic input ($\boldsymbol{w}_{k-1}$), the concatenation described previously is required. Accordingly, the augmented mean and covariance at time t_{k-1} are formed as

$$\breve{\boldsymbol{m}}^{+}_{k-1} = \begin{bmatrix} \boldsymbol{m}^{+}_{x,k-1} \\ \boldsymbol{0}_{n_w \times 1} \end{bmatrix} \qquad \text{and} \qquad \breve{\boldsymbol{P}}^{+}_{k-1} = \begin{bmatrix} \boldsymbol{P}^{+}_{xx,k-1} & \boldsymbol{0}_{n_x \times n_w} \\ \boldsymbol{0}_{n_w \times n_x} & \boldsymbol{P}_{ww,k-1} \end{bmatrix},$$

where is recalled that the process noise is taken to be zero-mean and uncorrelated with the state. Leveraging the concatenated mean and covariance, a set of q^{+}_{k-1} quadrature points and weights of the form $\breve{\boldsymbol{x}}^{+(\ell)}_{k-1}$, $w^{+(\ell)}_{m,k-1}$, and $w^{+(\ell)}_{c,k-1}$ are generated. The quadrature points are then partitioned into state and process noise points as

$$\breve{\boldsymbol{x}}^{+(\ell)}_{k-1} = \begin{bmatrix} \boldsymbol{x}^{+(\ell)}_{k-1} \\ \boldsymbol{w}^{(\ell)}_{k-1} \end{bmatrix}. \tag{2.18}$$

A set of transformed quadrature points is then formed by subjecting the concatenated, input quadrature points to the dynamics given by Eqs. (2.1), yielding

$$\boldsymbol{x}^{-(\ell)}_{k} = \boldsymbol{f}(\boldsymbol{x}^{+(\ell)}_{k-1}, \boldsymbol{w}^{(\ell)}_{k-1}),$$

and the predictor's expected values are given as

$$\boldsymbol{m}^{-}_{x,k} = \sum_{\ell=1}^{q^{+}_{k-1}} w^{+(\ell)}_{m,k-1} \boldsymbol{x}^{-(\ell)}_{k} \tag{2.19a}$$

$$\boldsymbol{P}^{-}_{xx,k} = \sum_{\ell=1}^{q^{+}_{k-1}} w^{+(\ell)}_{c,k-1} (\boldsymbol{x}^{-(\ell)}_{k} - \boldsymbol{m}^{-}_{x,k})(\boldsymbol{x}^{-(\ell)}_{k} - \boldsymbol{m}^{-}_{x,k})^{T}. \tag{2.19b}$$

The number of quadrature points is denoted by q^{+}_{k-1} to account for the general case where the number of points may be adapted through time.

2.1.5.2. Corrector. The corrector equations are obtained in much the same way as with the prediction stage. Augmented points are formed with the *a priori* mean and the measurement noise, producing the augmented mean and covariance

$$\breve{\boldsymbol{m}}_k^- = \begin{bmatrix} \boldsymbol{m}_{x,k}^- \\ \boldsymbol{0}_{n_v \times 1} \end{bmatrix} \qquad \text{and} \qquad \breve{\boldsymbol{P}}_k^- = \begin{bmatrix} \boldsymbol{P}_{xx,k}^- & \boldsymbol{0}_{n_x \times n_v} \\ \boldsymbol{0}_{n_v \times n_x} & \boldsymbol{P}_{vv,k} \end{bmatrix}$$

where the measurement noise is taken to be zero mean and uncorrelated with the state. A set of q_k^- quadrature points and weights of the form $\breve{\boldsymbol{x}}_k^{-(\ell)}$, $w_{m,k}^{-(\ell)}$, and $w_{c,k}^{-(\ell)}$ is generated and then partitioned according to

$$\breve{\boldsymbol{x}}_k^{-(\ell)} = \begin{bmatrix} \boldsymbol{x}_k^{-(\ell)} \\ \boldsymbol{v}_k^{(\ell)} \end{bmatrix} .$$

The *a priori* quadrature points are transformed through the nonlinear measurement function given by Eq. (2.2) to produce measurement quadrature points of the form

$$\boldsymbol{z}_k^{-(\ell)} = \boldsymbol{h}(\boldsymbol{x}_k^{-(\ell)}, \boldsymbol{v}_k^{(\ell)}) .$$

The expectations necessary for the update are computed as

$$\boldsymbol{m}_{z,k}^- = \sum_{\ell=1}^{q_k^-} w_{m,k}^{-(\ell)} \boldsymbol{z}_k^{-(\ell)} \tag{2.20a}$$

$$\boldsymbol{P}_{zz,k}^- = \sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)} (\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)^T \tag{2.20b}$$

$$\boldsymbol{P}_{xz,k}^- = \sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)} (\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^-)(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)^T , \tag{2.20c}$$

and the update is completed with Eqs. (2.6) and (2.8). As with the propagation, the number of quadrature points in the update, q_k^-, is permitted to vary through time. More importantly, the number of quadrature points used in the propagation stage is not at all required to be the same as the number of quadrature points used in the update.

Summary. The quadrature-based formulation of the MMSE filter is summarized in Table 2.3.

Table 2.3. Quadrature-based formulation of the MMSE filter.

Models	$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1})$ $\quad$ $\boldsymbol{P}_{ww,k-1} = \mathbb{E}\left\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\right\}$ $\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k)$ $\quad$ $\boldsymbol{P}_{vv,k} = \mathbb{E}\left\{\boldsymbol{v}_k\boldsymbol{v}_k^T\right\}$
Initialization	$\boldsymbol{m}_{x,0} = \mathbb{E}\left\{\boldsymbol{x}_0\right\}$ $\quad$ $\boldsymbol{P}_{xx,0} = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})^T\right\}$
Predictor	Draw points: $\boldsymbol{x}_{k-1}^{+(\ell)}$ and $\boldsymbol{w}_{k-1}^{(\ell)}$ with weights $w_{m,k-1}^{+(\ell)}$, $w_{c,k-1}^{+(\ell)}$ for $\ell \in \{1, \ldots, q_{k-1}^+\}$ Transform points: $\boldsymbol{x}_k^{-(\ell)} = \boldsymbol{f}(\boldsymbol{x}_{k-1}^{+(\ell)}, \boldsymbol{w}_{k-1}^{(\ell)})$ $\boldsymbol{m}_{x,k}^- = \sum_{\ell=1}^{q_{k-1}^+} w_{m,k-1}^{+(\ell)} \boldsymbol{x}_k^{-(\ell)}$ $\boldsymbol{P}_{xx,k}^- = \sum_{\ell=1}^{q_{k-1}^+} w_{c,k-1}^{+(\ell)} (\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^-)(\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^-)^T$
Corrector	Draw points: $\boldsymbol{x}_k^{-(\ell)}$ and $\boldsymbol{v}_k^{(\ell)}$ with weights $w_{m,k}^{-(\ell)}$, $w_{c,k}^{-(\ell)}$ for $\ell \in \{1, \ldots, q_k^-\}$ Transform points: $\boldsymbol{z}_k^{-(\ell)} = \boldsymbol{h}(\boldsymbol{x}_k^{-(\ell)}, \boldsymbol{v}_k^{(\ell)})$ $\boldsymbol{m}_{x,k}^+ = \boldsymbol{m}_{x,k}^- + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^-)$ $\boldsymbol{P}_{xx,k}^+ = \boldsymbol{P}_{xx,k}^- - \boldsymbol{P}_{xz,k}^- \boldsymbol{K}_{x,k}^T - \boldsymbol{K}_{x,k}(\boldsymbol{P}_{xz,k}^-)^T + \boldsymbol{K}_{x,k}\boldsymbol{P}_{zz,k}^- \boldsymbol{K}_{x,k}^T$ $\boldsymbol{K}_{x,k} = \boldsymbol{P}_{xz,k}^- (\boldsymbol{P}_{zz,k}^-)^{-1}$ $\boldsymbol{m}_{z,k}^- = \sum_{\ell=1}^{q_k^-} w_{m,k}^{-(\ell)} \boldsymbol{z}_k^{-(\ell)}$ $\boldsymbol{P}_{zz,k}^- = \sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)} (\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)^T$ $\boldsymbol{P}_{xz,k}^- = \sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)} (\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^-)(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)^T$

2.2. PRACTICAL NAVIGATION TECHNIQUES

As navigators began investigating the use of these MMSE filters in the 1960s, a common theme arose: numerical sensitivity causing filter failure. Even in ideal scenarios, filter divergence wreaked havoc upon these implementations, and a number of tools were developed to combat these effects. Many, if not most, of these early innovations are due to early manned spaceflight advancements, and it is the Apollo and Shuttle programs that bore the most lasting fruit. Despite being more than half a century since the wide dissemination of Kalman's seminal work, from a time where digital computing was in its infancy, it is the investigations and discoveries produced in the few years after the filter's introduction that are crucial in most real-world applications today.

It did not take long for practitioners to identify the value of Kalman's filtering methodologies, but it took just as short a time for the filter's unfortunate numerics to take its toll on computer implementations. Time and time again, numerical simulations produced incredibly poor solutions or faulted entirely due to filter divergence. It turned out that the causes of these effects could always be tied to at least one of two effects: (i) insufficient modeling of the system or (ii) computational error induced by roundoff from

finite precision arithmetic. The first of these effects can, at least, be treated by thorough consultation with a pen and paper, but roundoff error is an inevitable element of any computing procedure. The great minds working on these problems quickly devised a number of solutions to mitigate the effects of roundoff error, and it is these tools that has enabled the great human achievements in spaceflight (landing on the Moon, launching and landing Shuttle, operating a Global Positioning System (GPS) network, building the International Space Station, deploying rovers on Mars, nearly routinely launching and returning humans to and from life in orbit, etc.), ultimately propagating to other industries such as industrial automation, vehicle autonomy, medical imaging, and chemical laboratories, just to name a few.

Before any improvements were developed to mitigate filter failure, the principal cause was identified: loss of positivity of the state error covariance matrix. That is, the matrix $\boldsymbol{P}^{+}_{xx,k}$ fails to retain its positive-semidefinite property. When this property is lost, all subsequent computations become nonsensical and poorly conditioned, ultimately resulting in a failure to compute the required matrix inverse. Since this phenomenon is known to be caused by a lack of numerical precision, one must be cognizant of the cases that magnify the damaging effects of roundoff, and it is these cases that inspired the tools described in this section.

It should be noted that the widespread use of these tools does not mean that the problems of numerical sensitivity in sequential filtering has been completely obviated. These elements still plague practical applications to this very day, thus motivating and valorizing the developments of Section 3.

2.2.1. Residual Editing. Sometimes, sensors produce errant or faulty returns slated to be forwarded to and processed by the filter. If processed, these outliers can permanently degrade a filtering solution and, due to gross statistical disagreement with the employed models, promote divergence. A technique known as residual editing pre-qualifies incoming data to be processed by the filter or, instead, discarded entirely. These techniques date to the earliest Kalman filter implementations, some notable early cases being Lear's 1965 dissertation [21], the culmination of his time shared between Purdue University and TRW Systems Group in Redondo Beach, and its use in Apollo's lunar module powered flight

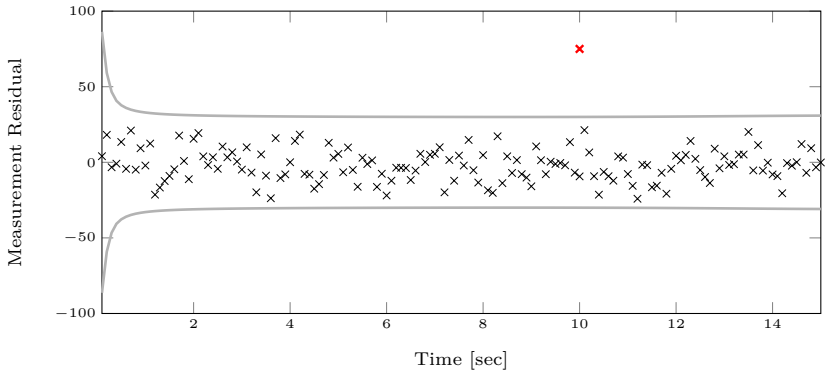


Figure 2.1. Illustrative use of residual editing to prevent processing an errant data return, with processed sensor returns as $\times$, edited return as $\times$, and 3σ residual covariance intervals as lines.

data processor [19].[4] The concept is simple: check for statistical disagreements between incoming data and the filter's model. One might be inclined to ask, "But what if the model is wrong and good data is edited out?" While this may be a reasonable question, it is most likely that an implementation with a faulty measurement model was doomed from the start.

An illustrative example of residual editing is shown in Figure 2.1. Here, the gray lines denote 3σ (that is, three times the standard deviation) intervals from the residual covariance, $\boldsymbol{P}_{zz,k}^{-}$. The sensor return denoted by $\times$ is clearly poorly represented by the *a priori* statistics and, therefore, is tagged and removed from candidacy for processing within the filter. If residual editing were not applied here, this errant return would have been processed by the filter and, at best, produced a poorly biased state estimate or, at worst and very likely, caused filter divergence.

There are a number of ways to perform residual editing, but two common methods are outlined presently.

2.2.1.1. Scalar editing. Recall that the measurement model is taken to be of the form

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_v)$$

[4]It is noted with respect to Ref. [19] that the first revision of this report was submitted on Christmas Eve, 1968 as Frank Borman, Jim Lovell, and Bill Anders of Apollo 8 entered lunar orbit.

such that the residual can be written as

$$\begin{aligned}\boldsymbol{r}_k^- &= \boldsymbol{z}_k - \mathbb{E}\left\{\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k)\right\} \\ &= \boldsymbol{z}_k - \boldsymbol{m}_{z,k}^-\end{aligned}$$

since the measurement noise $\boldsymbol{v}_k$ is taken to be zero-mean. Recalling that the filter presented earlier is unbiased, $\boldsymbol{r}_k^-$ is known to be zero-mean as well, i.e. $\mathbb{E}\left\{\boldsymbol{r}_k^-\right\} = \boldsymbol{0}_{n_z \times 1}$. The covariance of $\boldsymbol{r}_k$ is formally computed by the filter as the term $\mathbb{E}\left\{\boldsymbol{r}_k^-(\boldsymbol{r}_k^-)^T\right\} = \boldsymbol{P}_{zz,k}^-$ and is called the residual covariance.[5]

Let $r_{k,i}^-$ and $P_{zz,k,ii}^-$ denote the i^{th} (scalar) element and (scalar) diagonal element of $\boldsymbol{r}_k^-$ and $\boldsymbol{P}_{zz,k}^-$, respectively. Additionally, let $\eta_{r_k,i}$ be a scalar confidence interval to edit at (i.e. $\eta_{r_k,i} = 3$ is a 3σ edit, $\eta_{r_k,i} = 4$ is a 4σ edit etc.). Then, the scalar residual editing rule is such that if

$$|r_{k,i}^-| > \eta_{r_k,i}\sqrt{P_{zz,k,ii}^-}$$

is satisfied for any $i \in \{1, \ldots, n_z\}$, where $|\cdot|$ denotes the absolute value, $\boldsymbol{z}_k$ is discarded and not processed. This test effectively determines if the residual exceeds the prescribed editing interval and, if so, removes it from consideration by deeming it statistically unlikely. Intuitively, the larger $\eta_{r_k,i}$ is, the less editing occurs and, potentially, bad data will be processed. Conversely, if $\eta_{r_k,i}$ is too small, good data is erroneously removed from consideration. Selection of these criteria is an important design task, but, fortunately, setting all $\eta_{r_k,i}$ to 5 or 6 tends to behave well, as residuals beyond 5σ and 6σ are rather suspicious returns, indeed. In fact, Lear suggested $\eta_{r_k,i} = 6$ in his multi-phase navigation design for the Space Shuttle [22], and the Orion flight processor employed $\eta_{r_k,i} = 5$ for Exploration Flight Test 1 [30].

[5] Some refer to $\mathbb{E}\left\{\boldsymbol{r}_k^-(\boldsymbol{r}_k^-)^T\right\}$ as the "innovations covariance" and $\mathbb{E}\left\{\boldsymbol{r}_k^+(\boldsymbol{r}_k^+)^T\right\}$ as the residual covariance. This terminology is not employed here.

Practically speaking, if $\boldsymbol{z}_k$ contains simultaneous measurements from multiple sources (such as lidar information and magnetometer data), it logically extends to only entirely discard the edited sensor type. That is, if one of the lidar elements flags an edit check, one would discard all of the lidar information within $\boldsymbol{z}_k$ but still process the magnetometer data.

Remark 2.1. *There may be some disagreement about discarding all of $\boldsymbol{z}_k$ if, say, only one or two of $r^-_{k,i} \in \boldsymbol{r}^-_k$ fail an editing check (i.e. are too large). If it was so desired, one could simply remove the i^{th} element and diagonal entry (or entries) from $\boldsymbol{r}^-_k$ and $\boldsymbol{P}^-_{zz,k}$ and then process the remaining elements. However, it is argued here that if any of the $r^-_{k,i}$ arouse suspicion, it is safer to discard all of $\boldsymbol{z}_k$ than to process the attached, potentially erroneous, data. If an apple is found to be poisoned, it is sometimes best to avoid the whole bushel.*

A valid criticism of this scalar editing procedure is that it disregards cross-correlations within the residual covariance. Most of the time, this theoretical limitation is ignored with few ill-effects, such as in Shuttle navigation [31] or the Orion flight software [30]. If desired, one way of accounting for such correlations is via whitening, a method for de-correlating data, but this adds computational burden and is often skipped. Instead, one might consider the use of a method that formally accounts for these correlations from the beginning, here referred to as vector editing.

2.2.1.2. Vector editing. By defining the squared Mahalanobis distance

$$d^2 = (\boldsymbol{r}^-_k)^T (\boldsymbol{P}^-_{zz,k})^{-1} \boldsymbol{r}^-_k \,,$$

a vector-valued test function can be devised that has a convenient relationship with Gaussian statistics. Particularly, if the noise in $\boldsymbol{z}_k$ is taken to be additive and Gaussian, it is well known that d^2 then belongs to a χ^2 distribution with n_z degrees of freedom [32]. What this allows is for a user to define some probability gate, denoted as P_{edit}, and look up the corresponding value from a χ^2 table. These tables are widely distributed and are easy to find, such as in most statistics textbooks, and so one is not repeated here. Then, the editing

rule to check for disagreement is given as

$$d^2 > \eta(n_z, P_{\text{edit}}) \,,$$

where $\eta(\cdot,\cdot)$ now represents the table lookup. So, for a given $\boldsymbol{z}_k$, if $d^2 > \eta(n_z, P_{\text{edit}})$, $\boldsymbol{z}_k$ is rejected. As an example, if $n_z = 3$ and one selects $P_{\text{edit}} = 95\%$ such that only data agreeing statistically to 95% is processed, the table lookup yields $\eta(n_z, P_{\text{edit}}) = 7.81$.

This method comes with the burden of an additional modeling assumption that the measurement noise is Gaussian distributed, but offers a flexible and easy-to-interpret editing scheme that naturally captures correlations within the residual covariance. Fortunately, the assumption of Gaussianity is often a good one. It is important to note, however, that the vector editing scheme requires a matrix inversion that may impose additional computational burden such that the scalar editing method is more attractive.

2.2.2. Underweighting. The classes of functions that tend to characterize the natural motion of the universe and the processes by which useful measurement data are generated tend to be remarkably complex. Accordingly, the physics-based mathematical models that are concocted to emulate these phenomena tend to be nonlinear, and treating these nonlinearities was discussed in Section 2.1.4 for linearization-driven approaches and in Section 2.1.5 for quadrature-based approaches. Linearization is far and away the more commonly used methodology and remains the backbone of most real-world navigation applications. Part of the reason for this is certainly the historical significance of these techniques (they did serve as a cornerstone of the project that put American boots on the Moon, after all). Another reason could be that, in some sense, Taylor series is to an engineer what a hammer is to a mason.

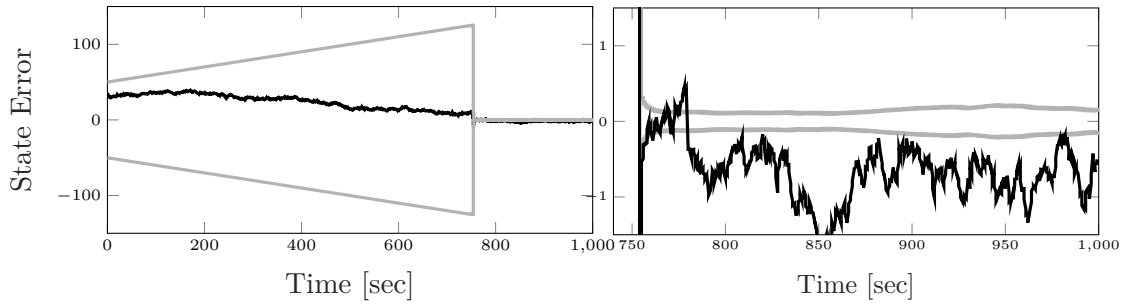
Whatever the reason, linearization has received the most attention in practical applications, and so Taylor series is the principal tool for navigators to assess errors in their own approximations of nonlinear systems. Ultimately, the result of these approximation errors is an under-estimation of the error covariance matrix, yielding an overconfident or "smug" filter. A smug filter latches on to an incorrect state solution and begins to disregard incoming data, regardless of the size of subsequent residuals, or it will produces

accurate state solutions with underestimated error statistics. Even if a filter is able to avoid being smug, processing precise measurements (very small numbers) against large *a priori* uncertainties (very large numbers) often results in poorly conditioned matrices and loss of covariance positivity. This is a particularly frustrating component in spaceflight because it is not uncommon to be without navigation data for extended periods of time, during which state uncertainties tend to grow due to uncertain and noisy dynamics. Higher order filtering solutions exist, such as the second order filter described by Athans [33] and Jazwinski [34], but are unattractive due to their required computational burden.

An illustrative example of the effects of underweighting is presented in Figure 2.2. In this figure, state errors are plotted as black lines, and the 1σ intervals from covariance that represent state uncertainty are plotted as gray lines. Recall that, in general, these intervals are interpreted as encapsulating or enveloping the state error, and consistent violations of this boundary indicate poor convergence.[6] In this case, a long 750 seconds worth of predictions is required before measurement data are obtained, and, due to relatively precise measurements, filter divergence is probable. Here, the top row of figures illustrates the case where no underweighting is employed, and, quite plainly, the filter converges upon an incorrect solution. The bottom row, however, utilizes underweighting, and this faulty convergence is prevented entirely. Note that the state uncertainty gradually decreases, due to the "softening" effect of underweighting, rather than the severe updates causing divergence in the top row.

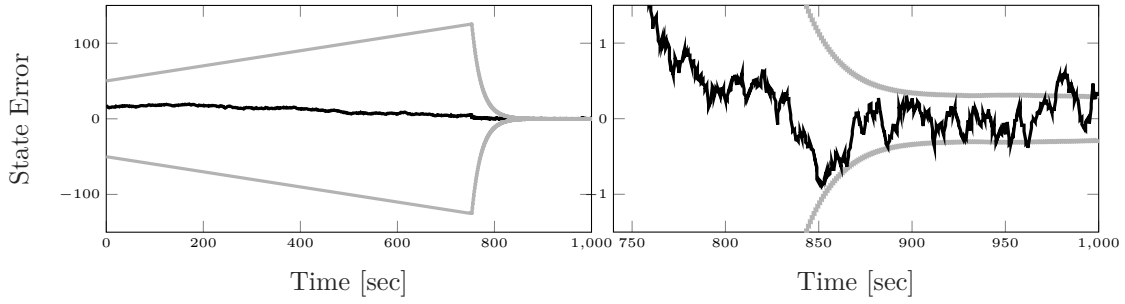
Remark 2.2 (An Example of Filter Degeneracy)**.** *To illustrate why these filters fail in the presence of large a priori uncertainties and accurate measurements, consider the following. Let the state be an uncertain scalar (i.e.* $\boldsymbol{x}_k = x_k$*) that is measured directly (i.e.* $\boldsymbol{H}_{x,k} = 1$*). Additionally, let the a priori variance be large with respect to the measurement noise (i.e* $\boldsymbol{P}_{xx,k} \gg \boldsymbol{P}_{vv,k}$*, or, for the scalar system,* $p^-_{xx,k} \gg p_{vv,k}$*) Then, the covariance update of*

[6] Typically, violations of a 3σ interval are deemed unacceptable due to the 3σ, or 99.7%, rule for Gaussians, but in the absence of truly Gaussian statistics, this is only approximate. Here, 1σ is plotted to illustrate trends; there is a plain bias in the converged-upon solution.



(a) No underweighting applied.

(b) No underweighting applied, zoomed.



(c) Underweighting applied.

(d) Underweighting applied, zoomed.

Figure 2.2. Illustration of the use of underweighting to prevent over-convergence upon an incorrect filtering solution where error is plotted in black and the 1σ intervals from the estimated covariance are plotted in gray.

Eq. (2.11) becomes a scalar equation of the form

$$\begin{aligned} \boldsymbol{P}^{+}_{xx,k} &= (\boldsymbol{I}_{n_x \times n_x} - \boldsymbol{K}_{x,k}\boldsymbol{H}_{x,k})\boldsymbol{P}^{-}_{xx,k} \\ \implies p^{+}_{xx,k} &= (1 - k_{x,k})p^{-}_{xx,k}\,. \end{aligned}$$

Suppose that some roundoff error is induced in the computation of the update gain and denote it by ϵ (i.e the numerical value of $k_{x,k}$ is actually $k_{x,k} + \epsilon$). Let this error be some small, positive value such that $\epsilon > 0$. Then, the processed update is actually given by

$$p^{+}_{xx,k} = (1 - k_{x,k} - \epsilon)p^{-}_{xx,k}\,.$$

As the a priori state is uncertain, the prior error variance is certainly greater than zero, i.e. $p^-_{xx,k} > 0$*, and the posterior error variance is at least nonnegative,*[7] *therefore*

$$\begin{aligned} 1 - k_{x,k} - \epsilon &\geq 0 \\ k_{x,k} + \epsilon &\leq 1\,. \end{aligned} \tag{2.21}$$

It's easy to show that, with Eq. (2.10) and Eqs. (2.16b)–(2.16c), the scalar update gain is given for this problem as

$$k_{x,k} = \frac{p^-_{xx,k}}{p^-_{xx,k} + p_{vv,k}} \tag{2.22}$$

Now, if $p^-_{xx,k} \gg p_{vv,k}$ *as was supposed, then* $k_{x,k} \approx 1$*. This implies that, from Eq. (2.21),*

$$\epsilon \leq 0\,,$$

which is, of course, not true. Therefore, $p^+_{xx,k}$ *must be negative, and the filtering solution becomes degenerate.*

This is just one example where roundoff errors caused by finite precision arithmetic can cause loss of covariance positivity for even the simplest problems. In particular, this illustrates a case where large a priori uncertainties and precise measurements promote filter degeneracy. What, then, can a practicing engineer hope to expect from the much more complicated problems involved in spaceflight?

Lear noticed this effect and published a 1973 memo at NASA that proposed an all-in-one, launch-to-landing "multi-phase navigation program" for the Space Shuttle, complete with theoretical studies and code [22]. Beyond innovative for its time, it is this little-known work that set the foundation for so many navigation principles that are common practice today. Among these is that of measurement underweighting, a method of "softening" the corrector stage of the filter such that it is hesitant to make drastic changes to the state

[7] The case that $p^+_{xx,k} = 0$ is a very unique case, indeed, and should be earnestly contemplated. Of what validity or usefulness is a claim of absolute certainty, that is, zero variance, to a filter?

solution and associated error covariance estimate. Its effect is that of inflating the posterior covariance, compensating for the missing terms that were truncated from the required Taylor series expansions, by deflating the update gain, $\boldsymbol{K}_{x,k}$.

There are many candidate methods for underweighting and it is not the focus of this dissertation, so only one is discussed here. If a reader is interesting in delving further, the work of Zanetti et al. in [35] is a thorough look into the importance and techniques of underweighting.

It is clear that "decreasing" the update gain results in a larger posterior covariance, but how specifically to make this change is what remains. Recall that the gain can be written as

$$\boldsymbol{K}_{x,k} = \boldsymbol{P}^{-}_{xz,k}(\boldsymbol{P}^{-}_{zz,k})^{-1}\,,$$

and substituting for the linearized approximations of the cross and residual covariance yields

$$\boldsymbol{K}_{x,k} = \boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k}\left(\boldsymbol{H}_{x,k}\boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k} + \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}^{T}_{v,k}\right)^{-1}\,.$$

Then, define an underweighted gain as (adorning $\boldsymbol{K}_{x,k}$ with a tilde to denote that it is underweighted)

$$\tilde{\boldsymbol{K}}_{x,k} = \boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k}\left(\boldsymbol{H}_{x,k}\boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k} + \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}^{T}_{v,k} + \boldsymbol{U}_{k}\right)^{-1}\,,$$

where $\boldsymbol{U}_k$ is some symmetric, positive-definite underweighting factor. This procedure will always produce a "smaller" update gain and, accordingly, a weaker update.

Define a parameter p, such that $0 < p < 1$, and let

$$\boldsymbol{U}_{k} = \frac{1-p}{p}\boldsymbol{H}_{x,k}\boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k}\,,$$

such that

$$\begin{aligned}\tilde{\boldsymbol{K}}_{x,k} &= \boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k}\left(\boldsymbol{H}_{x,k}\boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k}+\boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}^{T}_{v,k}+\frac{1-p}{p}\boldsymbol{H}_{x,k}\boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k}\right)^{-1}\\ &= \boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k}\left(\frac{1}{p}\boldsymbol{H}_{x,k}\boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k}+\boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}^{T}_{v,k}\right)^{-1}.\end{aligned}$$

Therefore, p determines the underweighting scheme and has an intuitive interpretation. The parameter p governs "how much" the incoming measurement data are to be trusted, and the deflation of the gain is done according to the mapped *a priori* uncertainty. The case $p \to 1$ approaches the standard filter, whereas $p \to 0$ approaches a filter that performs no update at all. Selection of p is a design task for a specific implementation, but Lear suggested $p = \frac{5}{6}$ (such that $\frac{1}{p} = 1.2$) for the Space Shuttle [22] and, indeed, it is this parameter that was ultimately adopted [31].

With a scheme defined, a rule should be concocted to select when to use the optimal gain, $\boldsymbol{K}_{x,k}$, and when to use the underweighted gain, $\tilde{\boldsymbol{K}}_{x,k}$. One such rule is to apply underweighting if

$$||\boldsymbol{H}_{x,k}\boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k}|| \geq \frac{p}{1-p}||\boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}^{T}_{v,k}||\,,$$

where $||\cdot||$ denotes the matrix norm. This rule is motivated by checking how much "larger" the mapped state uncertainty ($\boldsymbol{H}_{x,k}\boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k}$) is with respect to the mapped measurement uncertainty ($\boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}^{T}_{v,k}$). For example, if $p = \frac{5}{6}$, underweighting is applied if the mapped state uncertainty is at least 5 times larger than the mapped measurement uncertainty.

Remark 2.3 (Revisiting the Example)**.** *Observe that if Eq. (2.22) were replaced with an underweighted gain such that, selecting $p = \frac{5}{6}$ as an example,*

$$\tilde{k}_{x,k} = \frac{p^{-}_{xx,k}}{1.2p^{-}_{xx,k}+p^{-}_{vv,k}} \approx \frac{p^{-}_{xx,k}}{1.2p^{-}_{xx,k}} = \frac{5}{6} < 1\,,$$

and positivity of $p^{+}_{xx,k}$ is enforced (unless ϵ is as large as $\frac{1}{6}$, which would be absurd).

2.2.3. Brute-Force Symmetrization. Residual editing and underweighting seek to add robustness by overcoming theoretical challenges imposed by real data sources and linearization errors, whereas brute-force symmetrization is a technique that seeks to overcome the induced roundoff error directly. While a covariance matrix should always be symmetric, sometimes the effects of finite precision arithmetic manifest themselves as disagreement in off-diagonal elements. Some early methods computed $\boldsymbol{P}^{-}_{xx,k}$ and overwrote half of the off-diagonals with the other. Some very specialized implementations only computed the upper or lower triangular components directly. Many observed very early on, however, that simply adding the posterior covariance matrix with its transpose and dividing by two could offer improvements in filter stability; that is, overwriting the stored variable for $\boldsymbol{P}^{+}_{xx,k}$ such that

$$\boldsymbol{P}^{+}_{xx,k} \leftarrow \frac{1}{2}\left(\boldsymbol{P}^{+}_{xx,k} + (\boldsymbol{P}^{+}_{xx,k})^{T}\right) .$$

While this is a trivial expression analytically, since $\boldsymbol{P}^{+}_{xx,k} = (\boldsymbol{P}^{+}_{xx,k})^{T}$, this brute-force technique effectively "averages out" the roundoff errors induced by finite precision computing. This very minor modification offers additional protection from roundoff errors at virtually no cost to a processor. In fact, the operation is so cheap that a user could consider doing the same to the *a priori* covariance after prediction as well, i.e.

$$\boldsymbol{P}^{-}_{xx,k} \leftarrow \frac{1}{2}\left(\boldsymbol{P}^{-}_{xx,k} + (\boldsymbol{P}^{-}_{xx,k})^{T}\right) .$$

2.2.4. Process Noise Tuning. Another technique that finds extensive use in practical application is that of process noise tuning. In a similar vein to underweighting, which attempts to account for underestimated covariance matrices by softening the corrector step, process noise tuning inflates the state covariance to account for unmodeled interactions and missing terms from truncated Taylor series. In a linearization-based application, the tuned *a priori* covariance looks something like

$$\boldsymbol{P}^{-}_{xx,k} = \boldsymbol{F}_{x,k-1}\boldsymbol{P}^{+}_{xx,k-1}\boldsymbol{F}^{T}_{x,k-1} + \boldsymbol{F}_{w,k-1}\boldsymbol{P}_{ww,k-1}\boldsymbol{F}^{T}_{w,k-1} + \boldsymbol{Q}_{\text{tune}} ,$$

where $\boldsymbol{Q}_{\text{tune}}$ is some symmetric, positive-semidefinite $n_x \times n_x$ tuning matrix. It is the entries of this tuning matrix that govern additional growth in $\boldsymbol{P}_{xx,k}^{-}$ as it temporally evolves, serving as a fading-memory factor that prevents the filter from becoming smug and inflexible. Selecting $\boldsymbol{Q}_{\text{tune}}$ too large, however, can drastically degrade filtering performance and even promote filter divergence. As with any of these practical navigation techniques, properly tuned process noise can be the difference between a successful and unsuccessful implementation

A difficulty in practical implementation is that, in truth, $\boldsymbol{P}_{ww,k-1}$ never known to absolute certainty, and therefore $\boldsymbol{Q}_{\text{tune}}$ serves to help account for underestimation of $\boldsymbol{P}_{ww,k-1}$. The subtleties and details of tuning process noise could entertain discussion for an eternity, and many consider it an art that is only learned through years of experience. In the interest of not distracting this dissertation's theme, the discussion ends here. It deserves mention here because no summary of navigation techniques would be remotely complete without mention of process noise tuning.

Remark 2.4 (Process noise versus tuning noise)**.** *It is common to misrepresent the concepts of process noise and tuning noise. Process noise is a physical phenomena of a dynamical system and is not a knob to turn and tune a filter. Any changes in* $\boldsymbol{P}_{ww,k-1}$ *that differ from the actual statistics of* $\boldsymbol{w}_{k-1}$ *yield a flawed model. The tuning matrix* $\boldsymbol{Q}_{\text{tune}}$*, on the other hand, is a voluntary modification of the estimated statistics to overcoming shortcomings induced by modeling errors and numerical imprecision. The subtle distinction of the two is important; some things are mutable, like the tuning matrix, and others are elements of the universe out of the navigator's control.*

2.2.5. Estimating Attitude. Most navigation tasks are inexorably committed to estimating the full pose of a vehicle, i.e. its position, velocity, and attitude. When working with the rectilinear states of position and velocity, their convenient Cartesian domains make operations such as addition and subtraction well-defined and simple to work with. Attitude, on the other hand, belongs to the realm of spherical coordinates and, therefore, is not closed under the addition operations required by the previously described filter. It is tempting at first to adopt the highly-intuitive, three-parameter Euler-angle representation of attitude

and estimate those three parameters in a filter, regardless of the required additions and subtractions being theoretical violations. Inconveniently, however, all three-parameter attitude representations are necessarily singular, and, thus, most navigation implementations abandon classical Euler-angle attitude representations in favor of four-parameter attitude quaternions because they are nonsingular. Some applications employ a three-parameter set known as modified Rodrigues parameters, despite their inherent singularity, because the singularity is trivially avoided, and estimating three, versus four, parameters provides a useful dimensional reduction (i.e. computational savings). The most popular result is a multiplicative, rather than additive, update to the attitude states.

It is almost always necessary to estimate attitude in a navigation application, but the specific techniques for doing so are omitted to substantially simplify the resulting expressions. This may at first seem to be an egregious omission, but it turns out that most standard practices for attitude estimation can be applied to the contents of this dissertation with little-to-no modification. Practical techniques for estimating attitude parameters are a well-studied and well-represented topic in literature, and the textbooks [36] and [37] serve as valuable compendiums of attitude filtering tools.

2.3. SQUARE-ROOT FILTERING

The previous section has repeatedly discussed the havoc numerical errors can wreak upon filtering implementations. To reiterate, the failures induced by numerical errors are principally due to the loss of the positive-definite property in the state error covariance. Many techniques, like those described in the previous section, have been devised to combat the loss of positivity, but they mostly require tuning of design variables and are *ad hoc* modifications of the filter itself. They do not directly address the root cause of the problem, and that is the covariance positivity constraint. Instead, an entirely new approach was born and remains one of the most important developments in practical navigation since the Kalman filter's introduction. This approach is known as square-root filtering.

While working at the Instrumentation Laboratory at MIT in 1962, James Potter was tasked by Richard Battin to conceive of a solution to the covariance positivity problems they were experiencing in Apollo software for sextant-based navigation [38]. Potter took the weekend to delve into this, and he returned on Monday with a complete re-imagination of the problem that would have lasting impacts on the fields estimation and navigation [39].[8] The key to Potter's approach was, rather than estimating a covariance matrix that is subject to a positivity constraint, to estimate the square-root factor of the covariance matrix [41]. Then, if the software needed a covariance matrix at any time, one could simply form it from the estimated square-root factor via matrix multiplication.

Potter's technique was limited by the lack of inclusion of process noise (largely considered a constraint resulting from limited computational capabilities) and the requirement that measurements be processed as uncorrelated scalars, but the concept of square-root filtering became a permanently embedded research topic for practical filtering implementations. Potter's work with square-root factors was built upon by Schmidt [42] using orthogonalization techniques, and Carlson [43] introduced the use of triangular factors for a more efficient implementation using Householder's approach for triangularization of matrices [44]. These methods are all square-root filters in that they utilize some matrix-analog of the scalar square-root, and their advantages are that the stored square-root factor has entries that are closer to unity than its full covariance counterpart, thus effectively doubling the precision of the stored variable [45, 46]. A convenient technique for assessing how "poorly conditioned" a covariance matrix is is to compute its condition number, the ratio of its largest to its smallest eigenvalues. A very large condition number corresponds to large numerical errors in the related computations, and, in essence, square-root filters reduce the condition number substantially (to the square-root of that condition number).

The required square-root operations, however, consumed an inordinate proportion of available computational resources on processors of the time,[9] inspiring Bierman to develop a filtering recursion instead using the UDU factorization [47], and later Thornton developed

[8] In [15], Battin recounts that his earlier book [40] unfairly represented the value of Potter's square-root filter by banishing it to the exercises as Prob. 9.11. He goes on to mention that it certainly deserved greater prominence and cites it as of being utmost importance to the Apollo navigation system.

[9] Lear produced an excellent analysis of the computational costs of square-root filters in [24].

the temporal update for such a factorization [48]. The UDU method is "square-root free" in the sense that it does not require the square-roots along the diagonal of the triangular factor as in Carlson's method. The UDU factorization has proven a powerful tool for navigation applications and, in fact, finds use (as with consider filtering to be described in Section 3) in the NASA Orion navigation flight software [30, 49]. However, while mathematically elegant, and despite its benefits toward maintaining filter stability, the UDU factorization does not enjoy the same benefits of increase in precision as square-root methods as it is, true to its designation, square-root free. This work focuses on square-root filters for the benefits they offer with regard to numerical precision and due to their remarkable numerical stability. A reader interested in UDU filtering should reference Bierman's textbook in [50] for a wealth of information, experiential anecdotes, and commentary on techniques for practical filtering. The square-root operation has begun to be implemented in hardware on ("baked in" to) most every modern processing chip and is a much less costly operation than it once was. It remains a relevant concern for modern spaceflight, since mostly heritage (i.e. reliable but technologically outdated) processors are flown, but as space-capable computing improves and evolves, square-root filters have become increasingly attractive.

This section provides theoretical background on square-roots of symmetric, positive-definite matrices, describes the tools required to implement square-root filters, and then provides details on square-root filters for linearization- and quadrature-based implementations for nonlinear systems. These square-root formulations are not innovations of this dissertation, but they serve as springboards to the new developments of Section 3, wherein yet another technique for building stable filtering implementations, known as consider filtering, is formulated in a new, square-root form.

2.3.1. Preliminaries. This section details some necessary machinery that serves as the backbone of any square-root filter. The discussion aims to be as specific as possible, since small changes in formulation can yield certain "canned" numerical subroutines useless. For example, if one attempts to blindly apply widely distributed code libraries, such as those packaged with MATLAB, incorrect results may be produced since seemingly identical routines can, in fact, be very different in formulation (such as working with upper or lower

triangular matrix factors). Therefore, the reader is urged to exhibit careful use of these background elements, and, additionally, appropriate algorithms are distributed throughout this dissertation to aid in implementation of these techniques.

2.3.1.1. Square-root factors of positive definite matrices. For some Hermitian, positive-definite matrix $\boldsymbol{P}$, some matrix $\boldsymbol{S}$, called a Cholesky factor,[10] is guaranteed to exist such that [51]

$$\boldsymbol{P} = \boldsymbol{S}\boldsymbol{S}^T ,$$

where $\boldsymbol{S}$ is upper triangular. While this definition holds for complex matrices (with matrix transpose replaced with the conjugate transpose), this dissertation is only concerned with real matrices. Interestingly, the Cholesky factor of a positive-definite matrix is unique. The terminology "square-root factor" is employed as this definition serves as a matrix analog of the scalar square-root. Indeed, in the scalar case, the Cholesky factor reduces to the scalar (plus or minus) square-root.

Algorithm 1 presents a method for computing this Cholesky factor (adapted from the GAXPY formulation of Golub and van Loan [51] to produce an upper triangular factor such that $\boldsymbol{P} = \boldsymbol{S}\boldsymbol{S}^T$).

Algorithm 1 Computing Upper Triangular Cholesky Factor Such That $\boldsymbol{P} = \boldsymbol{S}\boldsymbol{S}^T$

function CHOL_UUT($\boldsymbol{P}$)
  $\boldsymbol{S} = \boldsymbol{P}$
  $\boldsymbol{S}$ is of dimension $n \times n$
  for $j = n : -1 : 1$ **do**
    if $j < n$ **then**
      $\boldsymbol{S}_{1:j,j} = \boldsymbol{S}_{1:j,j} - \boldsymbol{S}_{1:j,j+1:n}\boldsymbol{S}^T_{j,j+1:n}$
      $\boldsymbol{S}_{j+1:n,j} = \boldsymbol{0}_{(n-j)\times 1}$
    $\boldsymbol{S}_{1:j,j} = \boldsymbol{S}_{1:j,j}/\sqrt{\boldsymbol{S}_{j,j}}$
return $\boldsymbol{S}$

[10] This method is named for André-Louis Cholesky, a French artillery officer and mathematician in the early 20th century. This triangular factorization was developed to assist in solving geodesy problems that require the solution of normal equations for linear systems with least squares.

Remark 2.5. *Defining a factorization requires selecting conventions to define the factor and how it composes. For some positive-definite and symmetric* $\boldsymbol{P}$*, if* $\boldsymbol{U}_i$ *and* $\boldsymbol{L}_i$ *are upper and lower triangular matrices, respectively, one could desire*

$$\begin{aligned}\boldsymbol{P} &= \boldsymbol{U}_1\boldsymbol{U}_1^T \\ &= \boldsymbol{U}_2^T\boldsymbol{U}_2 \\ &= \boldsymbol{L}_1\boldsymbol{L}_1^T \\ &= \boldsymbol{L}_2^T\boldsymbol{L}_2\,.\end{aligned}$$

Therefore, there are four conceivable Cholesky-like, triangular factorizations that come to mind, but are they all equivalent? They all must be equivalent in some sense, of course, as clearly $\boldsymbol{U}_1\boldsymbol{U}_1^T = \boldsymbol{L}_2^T\boldsymbol{L}_2$ *implies* $\boldsymbol{U}_1 = \boldsymbol{L}_2^T$*, but how does it impact the final implementation? Are there any implications for choosing one over another? Instead of simply selecting a convention and moving on, it's important to give thought to what a certain convention affords you over another. It turns out that selecting the first convention (that is, selecting upper triangular factors such that* $\boldsymbol{P} = \boldsymbol{U}_1\boldsymbol{U}_1^T$*) is a crucial element of producing the efficient square-root consider filter implementations outlined later in Section 3.2.5 and this will be discussed there.*

Finally, two notes: (i) in most square-root filtering literature, the third convention, $\boldsymbol{P} = \boldsymbol{L}_1\boldsymbol{L}_1^T$*, is employed and (ii)* MATLAB*'s widely used* `chol.m` *employs the second convention,* $\boldsymbol{P} = \boldsymbol{U}_2^T\boldsymbol{U}_2$*. Again, this work employs the first,* $\boldsymbol{P} = \boldsymbol{U}_1\boldsymbol{U}_1^T$*.*

2.3.1.2. RQ factorization via Householder reflections. The discussion that follows this section extensively utilizes the RQ-decomposition of matrices. The upper triangular RQ-decomposition is employed here due to the prescribed form of square-root factors. Recall that this dissertation employs upper triangular factors such that $\boldsymbol{P} = \boldsymbol{S}\boldsymbol{S}^T$, and developments presented in later sections require decompositions of "fat" matrices, i.e. matrices that possess fewer rows than columns. Other factor designs (such as lower triangular,

compositions of the form $\boldsymbol{S}^T\boldsymbol{S}$, decompositions of "tall" matrices, etc.) might require a QR-decomposition with different triangular orientation. The QR- and RQ-decompositions are intimately related, but the distinction is made here for clarity.

In particular, some "fat" matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, where $m < n$, is decomposed into the upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ and orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$, such that

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0}_{m\times(n-m)} & \boldsymbol{R} \end{bmatrix} \boldsymbol{Q} = \bar{\boldsymbol{R}}\boldsymbol{Q}\,.$$

This decomposition finds usefulness in its connection to square-root factors of covariance matrices. Take the case where some symmetric, positive-definite matrix $\boldsymbol{P}$ can be written as $\boldsymbol{P} = \boldsymbol{A}\boldsymbol{A}^T$, where, again, $\boldsymbol{A}$ is a fat matrix. Then, an RQ-decomposition of $\boldsymbol{A}$ can be performed, such that

$$\boldsymbol{A}\boldsymbol{A}^T = \bar{\boldsymbol{R}}\boldsymbol{Q}(\bar{\boldsymbol{R}}\boldsymbol{Q})^T = \bar{\boldsymbol{R}}\boldsymbol{Q}\boldsymbol{Q}^T\bar{\boldsymbol{R}}^T = \bar{\boldsymbol{R}}\bar{\boldsymbol{R}}^T = \boldsymbol{R}\boldsymbol{R}^T\,.$$

Therefore, the matrix $\boldsymbol{R}$ is a valid square-root factor of $\boldsymbol{P}$. This indicates an interesting, and useful, conclusion: the product of a square-root factor and an orthogonal matrix (i.e. rotation or reflection matrix) is still a valid square-root factor of the same matrix.

This RQ-decomposition can be accomplished using a sequence of orthogonal Householder reflections of the form [44]

$$\boldsymbol{Q} = \boldsymbol{I} - \frac{2\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}}\,, \tag{2.23}$$

where $\boldsymbol{v}$ is known as the Householder vector, to produce a valid, upper-triangular square-root factor. There are many interesting elements of and uses for Householder reflections, but they are not discussed here; a curious reader is referred to [38, 44, 51]. In what follows, this matrix factorization is denoted in operator-form as $\boldsymbol{R} = \mathrm{rq}\{\boldsymbol{A}\}$, where only the upper-triangular component $\boldsymbol{R}$ is returned. This work suggests the use of Algorithm 2, adapted from Golub and Van Loan [51], for performing these reflections.

Packages from common code distributions, such as MATLAB's widely used `qr.m`, may not necessarily produce the desired result (upper triangular factors such that $\boldsymbol{R}\boldsymbol{R}^T$ produces the factored matrix). As such, the reader is warned to use caution when employing pre-packaged routines, and Algorithm 2 is included for clarity.

Algorithm 2 Householder Reflections for Fat Matrix Producing Upper Triangular Factor

function HR($\boldsymbol{A}$)
  $\boldsymbol{A}$ is of dimension $m \times n$
  for $i = 1, \ldots, m$ **do**
    Set $m_i = m - i + 1$ and $n_i = n - i + 1$
    $\boldsymbol{a} = \boldsymbol{A}_{m_i, 1:n_i}$ is row vector of dimension n_i
    $\sigma = \boldsymbol{a}_{1:n_i-1}\boldsymbol{a}_{1:n_i-1}^T$
    $\boldsymbol{v} = [a_1 \quad a_2 \quad \ldots \quad a_{n_i-1} \quad 1]$
    if $\sigma = 0$ **then**
      $\beta = 0$
    else
      $\mu = \sqrt{a_{n_i}^2 + \sigma}$
      if $a_{n_i} \leq 0$ **then**
        $v_{n_i} = a_{n_i} - \mu$
      else
        $v_{n_i} = -\sigma/(a_{n_i} + \mu)$
      $\beta = 2v_{n_i}^2/(\sigma + v_{n_i}^2)$
      $\boldsymbol{v} = \boldsymbol{v}/v_{n_i}$
    $\boldsymbol{A}_{1:m_i, 1:n_i} \leftarrow \boldsymbol{A}_{1:m_i, 1:n_i}[\boldsymbol{I}_{n_i \times n_i} - \beta(\boldsymbol{v}^T\boldsymbol{v})]$
return $\boldsymbol{R} = \boldsymbol{A}_{1:m, n-m+1:n}$

2.3.1.3. Cholesky updating and downdating. Many estimation problems require evaluating expressions of the form

$$\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{u}\boldsymbol{u}^T \,,$$

where $\boldsymbol{A}$ is symmetric, positive-definite and $\boldsymbol{u}$ is a vector of appropriate dimension. Substituting Cholesky factors such that $\boldsymbol{A} = \boldsymbol{S}\boldsymbol{S}^T$ and $\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{S}}\tilde{\boldsymbol{S}}^T$ yields

$$\tilde{\boldsymbol{S}}\tilde{\boldsymbol{S}}^T = \boldsymbol{S}\boldsymbol{S}^T + \boldsymbol{u}\boldsymbol{u}^T \,.$$

The idea is to perform the rank-1 modification of $\boldsymbol{u}\boldsymbol{u}^T$ to $\boldsymbol{S}\boldsymbol{S}^T$ and obtain $\tilde{\boldsymbol{S}}$ directly, without ever composing $\boldsymbol{S}\boldsymbol{S}^T$ or $\tilde{\boldsymbol{S}}\tilde{\boldsymbol{S}}^T$. This is a famous linear algebra structure known as the Cholesky update and is the backbone to the corrector stage of a square-root filter. More specifically, these filters employ what is somewhat playfully referred to as the Cholesky *downdate*, given by

$$\tilde{\boldsymbol{S}}\tilde{\boldsymbol{S}}^T = \boldsymbol{S}\boldsymbol{S}^T - \boldsymbol{u}\boldsymbol{u}^T \,,$$

where the subtraction, rather than the addition of the update, earns it its designation as a downdate.

A procedure to perform this update or downdate is shown in Algorithm 3. Here, an upper triangular Cholesky factor $\boldsymbol{S}$ such that $\boldsymbol{P} = \boldsymbol{S}\boldsymbol{S}^T$ is updated according to the rank-1 modification $\boldsymbol{u}\boldsymbol{u}^T$. The matrix $\boldsymbol{S}$ is overwritten in place, and the term γ determines if an update or a downdate is to be performed. For an update, set $\gamma = 1$; for a downdate, set $\gamma = -1$.

Algorithm 3 Rank-1 Cholesky Update ($\gamma = 1$) or Downdate ($\gamma = -1$)

function CHOLUP_UUT($\boldsymbol{S}, \boldsymbol{u}, \gamma$)
  $\boldsymbol{S}$ is of dimension $n \times n$
  for $j = n : -1 : 1$ **do**
    $r = \sqrt{S_{k,k}^2 + \gamma u_k^2}$
    $c = r / S_{k,k}$
    $s = u_k / S_{k,k}$
    $S_{k,k} = r$
    $\boldsymbol{S}_{1:k-1,k} = (\boldsymbol{S}_{1:k-1,k} + \gamma s \boldsymbol{u}_{1:k-1})/c$
    $\boldsymbol{u}_{1:k-1} = c\boldsymbol{u}_{1:k-1} - s\boldsymbol{S}_{1:k-1,k}$
return $\boldsymbol{S}$

Remark 2.6. *There is a close relationship between the Cholesky update and the RQ decomposition. If one factors the Cholesky update as*

$$\begin{aligned}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{S}}^T &= \boldsymbol{S}\boldsymbol{S}^T + \boldsymbol{u}\boldsymbol{u}^T \\ &= \begin{bmatrix}\boldsymbol{S} & \boldsymbol{u}\end{bmatrix}\begin{bmatrix}\boldsymbol{S}^T \\ \boldsymbol{u}^T\end{bmatrix},\end{aligned}$$

it is easy to see that the desired square-root factor is

$$\tilde{\boldsymbol{S}} = \begin{bmatrix} \boldsymbol{S} & \boldsymbol{u} \end{bmatrix} .$$

As this is non-square, an RQ-decomposition can be performed to obtain the desired triangular factor; that is,

$$\tilde{\boldsymbol{S}} = \mathrm{rq}\left\{ \begin{bmatrix} \boldsymbol{S} & \boldsymbol{u} \end{bmatrix} \right\} .$$

Again, this is because, as was observed, any orthogonal transformation of a square-root factor is still a valid square-root factor of the same matrix. This sequence of reflections can be performed with Householder reflections and Algorithm 1 via

$$\tilde{\boldsymbol{S}} = \mathrm{HR}\left\{ \begin{bmatrix} \boldsymbol{S} & \boldsymbol{u} \end{bmatrix} \right\} .$$

Therefore, the Cholesky update and the RQ-decomposition are, in some sense, equivalent for updates.

2.3.2. Linearization-based Approach. In the following, the linearization-based MMSE filtering scheme described in Section 2.1.4 is reformulated to replace all covariance matrices, e.g. $\boldsymbol{P}^{+}_{xx,k-1}$, $\boldsymbol{P}_{ww,k-1}$, $\boldsymbol{P}^{-}_{xx,k}$, etc., with their square-root factor counterparts, and these will be denoted in the same fashion, e.g. $\boldsymbol{S}^{+}_{xx,k-1}$, $\boldsymbol{S}_{ww,k-1}$, $\boldsymbol{S}^{-}_{xx,k}$, etc. That is, the notation follows exactly as one might expect: $\boldsymbol{S}^{+}_{xx,k}$ is the upper triangular, posterior square-root factor of the state error covariance such that $\boldsymbol{P}^{+}_{xx,k} = \boldsymbol{S}^{+}_{xx,k}(\boldsymbol{S}^{+}_{xx,k})^T$. In like fashion to the full covariance formulation, it is assumed that the filter is initialized with some initial mean, $\boldsymbol{m}_{x,0}$, and square-root factor, $\boldsymbol{S}_{xx,0}$.

2.3.2.1. Predictor. The prediction for the covariance matrix is given in Eq. (2.15b) and is repeated here as

$$\boldsymbol{P}^{-}_{xx,k} = \boldsymbol{F}_{x,k-1}\boldsymbol{P}^{+}_{xx,k-1}\boldsymbol{F}^{T}_{x,k-1} + \boldsymbol{F}_{w,k-1}\boldsymbol{P}_{ww,k-1}\boldsymbol{F}^{T}_{w,k-1} .$$

Substituting in for square-root factors yields

$$\boldsymbol{S}^{-}_{xx,k}(\boldsymbol{S}^{-}_{xx,k})^T = \boldsymbol{F}_{x,k-1}\boldsymbol{S}^{+}_{xx,k-1}(\boldsymbol{S}^{+}_{xx,k-1})^T\boldsymbol{F}^T_{x,k-1} + \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1}\boldsymbol{S}^T_{ww,k-1}\boldsymbol{F}^T_{w,k-1}\,.$$

This can be factored as

$$\boldsymbol{S}^{-}_{xx,k}(\boldsymbol{S}^{-}_{xx,k})^T = \left[\begin{array}{c|c} \boldsymbol{F}_{x,k-1}\boldsymbol{S}^{+}_{xx,k-1} & \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1} \end{array}\right] \left[\begin{array}{c} (\boldsymbol{S}^{+}_{xx,k-1})^T\boldsymbol{F}^T_{x,k-1} \\ \boldsymbol{S}^T_{ww,k-1}\boldsymbol{F}^T_{w,k-1} \end{array}\right],$$

and comparing terms permits the conclusion that

$$\boldsymbol{S}^{-}_{xx,k} = \left[\begin{array}{c|c} \boldsymbol{F}_{x,k-1}\boldsymbol{S}^{+}_{xx,k-1} & \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1} \end{array}\right] \in \mathbb{R}^{n_x\times(n_x+n_w)}\,. \tag{2.24}$$

This is, in fact, a valid expression for the predicted square-root factor, but it is not the *desired* predicted square-root factor. This is due to the non-square nature of the predicted factor, and this served as a point of puzzlement for early square-root filtering applications (necessitating the removal of process noise from Potter's original formulation, interestingly). To further illustrate this point, note that if one were to immediately perform another prediction step to compute $\boldsymbol{S}^{-}_{xx,k+1}$, they would obtain

$$\boldsymbol{S}^{-}_{xx,k+1} = \left[\begin{array}{c|c|c} \boldsymbol{F}_{x,k}\boldsymbol{F}_{x,k-1}\boldsymbol{S}^{+}_{xx,k-1} & \boldsymbol{F}_{x,k}\boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1} & \boldsymbol{F}_{w,k}\boldsymbol{S}_{ww,k} \end{array}\right] \in \mathbb{R}^{n_x\times 2(n_x+n_w)}\,.$$

While is this a valid square-root factor of $\boldsymbol{P}^{-}_{xx,k+1}$, the progressive fattening makes implementation of such a filter entirely infeasible. Instead, the fact that any orthogonal transformation of a square-root factor is still a valid square-root factor of the same matrix can be used. To that end, if one identifies Eq. (2.24) as a fat matrix to be triangularized, the desired upper triangular, *a priori* square-root factor can be found using RQ-factorization, yielding

$$\boldsymbol{S}^{-}_{xx,k} = \mathrm{rq}\left\{\left[\begin{array}{c|c} \boldsymbol{F}_{x,k-1}\boldsymbol{S}^{+}_{xx,k-1} & \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1} \end{array}\right]\right\}$$

and can be computed with Algorithm 2 as

$$\boldsymbol{S}^{-}_{xx,k} = \text{HR}\left\{ \left[\begin{array}{c|c} \boldsymbol{F}_{x,k-1}\boldsymbol{S}^{+}_{xx,k-1} & \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1} \end{array} \right] \right\} . \tag{2.25}$$

The mean prediction is unchanged by any of these developments, i.e. it is still given by Eq. (2.15a), repeated here as

$$\boldsymbol{m}^{-}_{x,k} = \boldsymbol{f}(\boldsymbol{m}^{+}_{x,k-1}, \boldsymbol{0}_{n_w \times 1}) .$$

2.3.2.2. Corrector. With the *a priori* quantities obtained, the corrector step must be defined for square-root factors. Start by recalling that the update gain, the optimal Kalman gain, is given as

$$\boldsymbol{K}_{x,k} = \boldsymbol{P}^{-}_{xz,k}(\boldsymbol{P}^{-}_{zz,k})^{-1} ,$$

where $\boldsymbol{P}^{-}_{xz,k}$ and $\boldsymbol{P}^{-}_{zz,k}$ are the cross and residual covariances, respectively. Equation (2.16b) illustrated that the cross covariance term is computed as

$$\boldsymbol{P}^{-}_{xz,k} = \boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k} ,$$

and, substituting for the *a priori* square-root factor for the state error covariance, this can be written as

$$\boldsymbol{P}^{-}_{xz,k} = \boldsymbol{S}^{-}_{xx,k}[\boldsymbol{H}_{x,k}\boldsymbol{S}^{-}_{xx,k}]^{T} , \tag{2.26}$$

Further, recall that the residual covariance in Eq. (2.16c) is given as

$$\boldsymbol{P}^{-}_{zz,k} = \boldsymbol{H}_{x,k}\boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k} + \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}^{T}_{v,k} .$$

Substituting for square-root factors yields

$$\boldsymbol{S}^{-}_{zz,k}(\boldsymbol{S}^{-}_{zz,k})^{T} = \boldsymbol{H}_{x,k}\boldsymbol{S}^{-}_{xx,k}(\boldsymbol{S}^{-}_{xx,k})^{T}\boldsymbol{H}^{T}_{x,k} + \boldsymbol{H}_{v,k}\boldsymbol{S}_{vv,k}(\boldsymbol{S}_{vv,k})^{T}\boldsymbol{H}^{T}_{v,k} ,$$

which, following the same logic as in obtaining Eq. (2.25), can be factored to produce the desired square-root factor $\boldsymbol{S}_{zz,k}^{-}$ with Algorithm 2 as

$$\boldsymbol{S}_{zz,k}^{-} = \text{HR}\left\{ \left[\begin{array}{c|c} \boldsymbol{H}_{x,k}\boldsymbol{S}_{xx,k}^{-} & \boldsymbol{H}_{v,k}\boldsymbol{S}_{vv,k} \end{array} \right] \right\} . \tag{2.27}$$

Then, the update gain can be rewritten in terms of square-root factors as

$$\boldsymbol{K}_{x,k} = \boldsymbol{P}_{xz,k}^{-}(\boldsymbol{S}_{zz,k}^{-})^{-T}(\boldsymbol{S}_{zz,k}^{-})^{-1} ,$$

where $(\cdot)^{-T}$ denotes the inverse transposed and $\boldsymbol{P}_{xz,k}^{-}$ is computed as in Eq. (2.26). Define the term

$$\boldsymbol{U}_k \triangleq \boldsymbol{P}_{xz,k}^{-}(\boldsymbol{S}_{zz,k}^{-})^{-T} \tag{2.28}$$

such that the gain is given by

$$\boldsymbol{K}_{x,k} = \boldsymbol{U}_k(\boldsymbol{S}_{zz,k}^{-})^{-1} . \tag{2.29}$$

The purpose of this definition will become clear momentarily. Note that all required inverses thus far are very conveniently and reliably computed because they are inverses of triangular matrices, and Cholesky factors are always invertible. In fact, the triangular nature of these matrices entirely obviates the need for direct inversion entirely and instead can be solved as two nested inverse (least squares [52]) problems, the first being $\boldsymbol{A} = \boldsymbol{P}_{xz,k}^{-}(\boldsymbol{S}_{zz,k}^{-})^{-T}$ and the second being $\boldsymbol{K}_{x,k} = \boldsymbol{A}(\boldsymbol{S}_{zz,k}^{-})^{-1}$.

Recall that the posterior covariance was given in Eq. (2.8) as

$$\boldsymbol{P}_{xx,k}^{+} = \boldsymbol{P}_{xx,k}^{-} - \boldsymbol{P}_{xz,k}^{-}\boldsymbol{K}_{x,k}^{T} - \boldsymbol{K}_{x,k}(\boldsymbol{P}_{xz,k}^{-})^{T} + \boldsymbol{K}_{x,k}\boldsymbol{P}_{zz,k}^{-}\boldsymbol{K}_{x,k}^{T} ,$$

and, as described in Section 2.1.3, use of the optimal gain permits this to be rewritten as

$$\boldsymbol{P}_{xx,k}^{+} = \boldsymbol{P}_{xx,k}^{-} - \boldsymbol{K}_{x,k}\boldsymbol{P}_{zz,k}^{-}\boldsymbol{K}_{x,k}^{T} .$$

Employing the definition of Eq. (2.28) and substituting in for square-root factors permit this to be expressed as

$$\boldsymbol{S}^{+}_{xx,k}(\boldsymbol{S}^{+}_{xx,k})^T = \boldsymbol{S}^{-}_{xx,k}(\boldsymbol{S}^{-}_{xx,k})^T - \boldsymbol{U}_k\boldsymbol{U}_k^T\,,$$

which can be identified as a rank-n_z modification (downdate) of $\boldsymbol{S}^{-}_{xx,k}(\boldsymbol{S}^{-}_{xx,k})^T$. Letting $\boldsymbol{u}_i$ denote the i^{th} column of the matrix $\boldsymbol{U}_k$ allows the rank-n_z downdate to be rewritten as n_z rank-1 downdates of the form

$$\boldsymbol{S}^{+}_{xx,k}(\boldsymbol{S}^{+}_{xx,k})^T = \boldsymbol{S}^{-}_{xx,k}(\boldsymbol{S}^{-}_{xx,k})^T - \boldsymbol{u}_1\boldsymbol{u}_1^T - \ \ldots\ - \boldsymbol{u}_{n_z}\boldsymbol{u}_{n_z}^T\,.$$

These rank-1 downdates are precisely the Cholesky downdates described in Section 2.3.1.3 and can be applied sequentially using Algorithm 3. That is, set $\boldsymbol{S}^* = \boldsymbol{S}^{-}_{xx,k}$, sequentially perform

$$\boldsymbol{S}^* \leftarrow \text{CHOLUP\_UUT}\left\{\boldsymbol{S}^*, \boldsymbol{u}_i, -1\right\}\,, \qquad i \in \{1, \ldots, n_z\}\,,$$

and, once all n_z downdates are completed, set $\boldsymbol{S}^{+}_{xx,k} = \boldsymbol{S}^*$.

The state mean update is unaffected by these developments, but with the Kalman gain computed somewhat differently, and so the mean update is still given by Eq. (2.6), repeated here as

$$\boldsymbol{m}^{+}_{x,k} = \boldsymbol{m}^{-}_{x,k} + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}^{-}_{z,k})\,,$$

where $\boldsymbol{K}_{x,k}$ is computed using Eq. (2.29).

Summary. The linearization-based square-root filter is summarized in Table 2.4.

Remark 2.7. *This section has described how to update the a priori square-root factor using a sequence of rank-1 modifications, and Section 3.2.4 details a new way of handling the square-root factor correction step. Specifically, it presents a way to perform the full, rank-n_z modification to $\boldsymbol{S}^{-}_{xx,k}$ in one pass rather than performing a sequence of n_z rank-1 Cholesky downdates.*

Table 2.4. Linearization-based square-root formulation of the MMSE filter.

Models	$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1})$	$\boldsymbol{S}_{ww,k-1}\boldsymbol{S}_{ww,k-1}^T = \mathbb{E}\left\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\right\}$
	$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k)$	$\boldsymbol{S}_{vv,k}\boldsymbol{S}_{vv,k}^T = \mathbb{E}\left\{\boldsymbol{v}_k\boldsymbol{v}_k^T\right\}$
Initialization	$\boldsymbol{m}_{x,0} = \mathbb{E}\left\{\boldsymbol{x}_0\right\}$	$\boldsymbol{S}_{xx,0}\boldsymbol{S}_{xx,0}^T = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})^T\right\}$
Predictor	$\boldsymbol{m}_{x,k}^- = \boldsymbol{f}(\boldsymbol{m}_{x,k-1}^+, \boldsymbol{0}_{n_w\times 1})$	
	$\boldsymbol{S}_{xx,k}^- = \text{HR}\left\{\left[\ \boldsymbol{F}_{x,k-1}\boldsymbol{S}_{xx,k-1}^+ \ \middle|\ \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1}\ \right]\right\}$	
Corrector	$\boldsymbol{m}_{x,k}^+ = \boldsymbol{m}_{x,k}^- + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^-)$	
	$\boldsymbol{K}_{x,k} = \boldsymbol{U}_k(\boldsymbol{S}_{zz,k}^-)^{-1}$	
	$\boldsymbol{U}_k = \boldsymbol{P}_{xz,k}^-(\boldsymbol{S}_{zz,k}^-)^{-T}$	
	$\boldsymbol{m}_{z,k}^- = \boldsymbol{h}(\boldsymbol{m}_{x,k}^-, \boldsymbol{0}_{n_v\times 1})$	
	$\boldsymbol{S}_{zz,k}^- = \text{HR}\left\{\left[\ \boldsymbol{H}_{x,k}\boldsymbol{S}_{xx,k}^- \ \middle|\ \boldsymbol{H}_{v,k}\boldsymbol{S}_{vv,k}\ \right]\right\}$	
	$\boldsymbol{P}_{xz,k}^- = \boldsymbol{S}_{xx,k}^-[\boldsymbol{H}_{x,k}\boldsymbol{S}_{xx,k}^-]^T$	
	Set $\boldsymbol{S}^* = \boldsymbol{S}_{xx,k}^-$ and for $i \in \{1, \ldots, n_z\}$ compute	
	$\boldsymbol{S}^* \leftarrow \text{CHOLUP\_UUT}\left\{\boldsymbol{S}^*, \boldsymbol{u}_i, -1\right\}$	
	Set $\boldsymbol{S}_{xx,k}^+ = \boldsymbol{S}^*$	

Remark 2.8. *It should be noted that this square-root update relies on the use of the optimal gain $\boldsymbol{K}_{x,k}$. If underweighting or other suboptimal filtering strategies are employed, the theoretical requirements of this update are violated. Instead, one could rewrite Joseph's form of Eq. (2.13) in terms of square-root factors, factor results, and obtain the updated square-root factor as*

$$\boldsymbol{S}_{xx,k}^+ = \text{rq}\left\{\left[\ (\boldsymbol{I}_{n_x\times n_x} - \tilde{\boldsymbol{K}}_{x,k}\boldsymbol{H}_{x,k})\boldsymbol{S}_{xx,k}^- \ \middle|\ \tilde{\boldsymbol{K}}_{x,k}\boldsymbol{S}_{vv,k}\ \right]\right\},$$

where the gain is adorned with a tilde to note that it is potentially different than the optimal gain. This is easy to show and so proof is omitted here, but note that this can be accomplished with the routine $\text{HR}\{\cdot\}$ *in Algorithm 2.*

This square-root update has the advantage of accommodating tools, such as underweighting, since a suboptimal gain is permitted, but has the downside of potentially losing the the elegant "nested least squares" solution when computing $\boldsymbol{K}_k$. That said, most suboptimal rules are small modifications of the optimal Kalman gain computation, and, therefore, usually retain the desired nested solution format when computing the suboptimal gain $\tilde{\boldsymbol{K}}_{x,k}$.

2.3.3. Quadrature-based Approach. This section seeks to repeat the developments of the previous section but for square-root filters that employ quadrature-based approximation of the required expectations in the MMSE filter, and this discussion will leverage the developments of the quadrature-based full covariance formulation presented in Section 2.1.5. The resulting equations can be primarily traced to van der Merwe and Wan [52], and van der Merwe's dissertation in [53] serves as an excellent, and thorough, resource for these methods.

Square-root formulations of the quadrature-based filters are especially attractive because, in most cases, they are a more natural fit than full covariance formulations. What is meant by this is that most quadrature schemes (such as the now-ubiquitous unscented transform of Uhlmann [54] and Julier [28]) require computing the square-root factor of the covariance matrix to draw quadrature points. While well-defined, this can serve as a substantial computational burden since it is required at every predictor and corrector step. A square-root formulation of such a filter avoids explicitly computing this at all since the square-root factor is naturally stored by the filter, in addition to gleaning the tremendous stability benefits afforded by a square-root formulation.

It is assumed that the filter is initialized with some initial mean, $\boldsymbol{m}_{x,0}$, and square-root factor, $\boldsymbol{S}_{xx,0}$.

2.3.3.1. Predictor. As the state dynamics, given by Eq. (2.1), contain a stochastic input ($\boldsymbol{w}_{k-1}$), concatenation is required as it was in Section 2.1.5. However, square-root factors replace covariance matrices. Accordingly, the augmented mean and square-root factor at time t_{k-1} are formed as

$$\breve{\boldsymbol{m}}_{k-1}^{+} = \begin{bmatrix} \boldsymbol{m}_{x,k-1}^{+} \\ \boldsymbol{0}_{n_w \times 1} \end{bmatrix} \qquad \text{and} \qquad \breve{\boldsymbol{S}}_{k-1}^{+} = \begin{bmatrix} \boldsymbol{S}_{xx,k-1}^{+} & \boldsymbol{0}_{n_x \times n_w} \\ \boldsymbol{0}_{n_w \times n_x} & \boldsymbol{S}_{ww,k-1} \end{bmatrix},$$

where it is recalled that the process noise is taken to be zero-mean and uncorrelated with the state. Leveraging the concatenated mean and square-root factor, a set of q_{k-1}^{+} quadrature points and weights of the form $\breve{\boldsymbol{x}}_{k-1}^{+(\ell)}$, $w_{m,k-1}^{+(\ell)}$, and $w_{c,k-1}^{+(\ell)}$ are generated. As before, the number of quadrature points is denoted by q_{k-1}^{+} to account for the general case where the number of points may be adapted through time. The quadrature points are then partitioned

into state and process noise points as

$$\breve{\boldsymbol{x}}_{k-1}^{+(\ell)} = \begin{bmatrix} \boldsymbol{x}_{k-1}^{+(\ell)} \\ \boldsymbol{w}_{k-1}^{(\ell)} \end{bmatrix} . \tag{2.30}$$

A set of transformed quadrature points is then formed by subjecting the concatenated, input quadrature points to the system dynamics, yielding

$$\boldsymbol{x}_k^{-(\ell)} = \boldsymbol{f}(\boldsymbol{x}_{k-1}^{+(\ell)}, \boldsymbol{w}_{k-1}^{(\ell)}) .$$

The predictor's mean is unaffected by the square-root factor substitution and is given as

$$\boldsymbol{m}_{x,k}^{-} = \sum_{\ell=1}^{q_{k-1}^{+}} w_{m,k-1}^{+(\ell)} \boldsymbol{x}_k^{-(\ell)} ,$$

but the predicted square-root factor requires more consideration. The *a priori* state covariance is given in Eq. (2.19b) as

$$\boldsymbol{P}_{xx,k}^{-} = \sum_{\ell=1}^{q_{k-1}^{+}} w_{c,k-1}^{+(\ell)} (\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^{-})(\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^{-})^T ,$$

and substituting for the *a priori* square-root factor produces the expression

$$\boldsymbol{S}_{xx,k}^{-} (\boldsymbol{S}_{xx,k}^{-})^T = \sum_{\ell=1}^{q_{k-1}^{+}} w_{c,k-1}^{+(\ell)} (\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^{-})(\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^{-})^T .$$

Normally, this expression could be factored to obtain a rectangular matrix for $\boldsymbol{S}_{xx,k}^{-}$ and an RQ-decomposition could be used to triangularize the result into the desired square matrix. However, in general, it is possible that one or more of the q_{k-1}^{+} covariance weights, $w_{c,k-1}^{+(\ell)}$, is *negative*. If the above expression is naively factored, the result would be $\sqrt{w_{c,k-1}^{+(\ell)}}$ terms that, if negative, are imaginary! A different technique is required.

Define the weighting matrix

$$\boldsymbol{W} \triangleq \operatorname{diag} \left\{ \sqrt{|w_{c,k-1}^{+(1)}|}, \ldots, \sqrt{|w_{c,k-1}^{+(q_{k-1}^{+})}|} \right\} ,$$

where diag$\{\cdot\}$ denotes diagonal concatenation of the input arguments, and let $\boldsymbol{W}^{(i:j)}$ denote the (square) submatrix of $\boldsymbol{W}$ containing the i^{th} to j^{th} rows and columns of $\boldsymbol{W}$. Additionally, let $\boldsymbol{x}_k^{-(i:j)}$ denote a column-wise concatenation of the i^{th} to j^{th} quadrature points. Finally, let the weights be sorted such that

$$w_{c,k-1}^{+(1)}, \ldots, w_{c,k-1}^{+(q_{k-1}^+)} \rightarrow \begin{cases} w_{c,k-1}^{+(1)}, \ldots, w_{c,k-1}^{+(r)} & \text{are the } r \text{ negative weights} \\ w_{c,k-1}^{+(r+1)}, \ldots, w_{c,k-1}^{+(q_{k-1}^+)} & \text{are the } (q_{k-1}^+ - r) \text{ nonnegative weights} \end{cases} .$$

That is, with respect to the index ℓ, the first r weights are negative and the remaining weights are nonnegative.

Then, the *a priori* square-root factor can be found by first performing the RQ-decomposition (using Householder reflections with Algorithm 2)[11]

$$\begin{aligned} \boldsymbol{S}^* &= \text{rq}\left\{ (\boldsymbol{x}_k^{-(r+1:q_{k-1}^+)} \ominus \boldsymbol{m}_{x,k}^-)\boldsymbol{W}^{(r+1:q_{k-1}^+)} \right\} \\ &= \text{HR}\left\{ (\boldsymbol{x}_k^{-(r+1:q_{k-1}^+)} \ominus \boldsymbol{m}_{x,k}^-)\boldsymbol{W}^{(r+1:q_{k-1}^+)} \right\} \end{aligned}$$

and then a Cholesky downdate on the result (using Algorithm 3)

$$\boldsymbol{S}_{xx,k}^- = \text{CHOLUP\_UUT}\left\{ \boldsymbol{S}^*, (\boldsymbol{x}_k^{-(1:r)} \ominus \boldsymbol{m}_{x,k}^-)\boldsymbol{W}^{(1:r)}, -1 \right\} ,$$

yielding the *a priori* square-root factor. In the case that $r = 0$, the downdate is simply skipped and only the RQ-decomposition is required. Effectively, and RQ-decomposition is used to account for the positive weights, and a sequence of Cholesky downdates is used to account for the negative weights.

2.3.3.2. Corrector. First, new quadrature points, $\boldsymbol{x}_k^{-(\ell)}$ and $\boldsymbol{v}_k^{(\ell)}$, are obtained using the predicted mean and square root factor, and the measurement transformed quadrature points are obtained via

$$\boldsymbol{z}_k^{-(\ell)} = \boldsymbol{h}(\boldsymbol{x}_k^{-(\ell)}, \boldsymbol{v}_k^{(\ell)}) ,$$

[11] The shorthand notation $\boldsymbol{A} \ominus \boldsymbol{a}$ is used to indicate that vector $\boldsymbol{a}$ is subtracted from each column of matrix $\boldsymbol{A}$.

where it is assumed that the points have been sorted in the manner described in the previous section. Then, noting that, as in Section 2.3.2.2, the update gain can be expressed as

$$\boldsymbol{K}_{x,k} = \boldsymbol{P}^{-}_{xz,k}(\boldsymbol{S}^{-}_{zz,k})^{-T}(\boldsymbol{S}^{-}_{zz,k})^{-1}\,,$$

where, from the quadrature-based filter discussion of Section 2.1.1,

$$\begin{aligned}\boldsymbol{m}^{-}_{z,k} &= \sum_{\ell=1}^{q_k^-} w^{-(\ell)}_{m,k}\,\boldsymbol{z}^{-(\ell)}_k \\ \boldsymbol{P}^{-}_{xz,k} &= \sum_{\ell=1}^{q_k^-} w^{-(\ell)}_{c,k}(\boldsymbol{x}^{-(\ell)}_k - \boldsymbol{m}^{-}_{x,k})(\boldsymbol{z}^{-(\ell)}_k - \boldsymbol{m}^{-}_{z,k})^T\end{aligned}$$

and $\boldsymbol{S}^{-}_{zz,k}$ is to be determined presently. Note that the term r, the number of negative weights, may change for subsequent predictor/corrector steps and should be monitored/altered accordingly.

In the same way as the predictor, $\boldsymbol{S}^{-}_{zz,k}$ must be computed in two stages to account for negative weights, the first being the RQ-decomposition

$$\begin{aligned}\boldsymbol{S}^{\dagger} &= \mathrm{rq}\left\{(\boldsymbol{z}_k^{-(r+1:q_k^-)} \ominus \boldsymbol{m}^{-}_{z,k})\boldsymbol{W}^{(r+1:q_k^-)}\right\} \\ &= \mathrm{HR}\left\{(\boldsymbol{z}_k^{-(r+1:q_k^-)} \ominus \boldsymbol{m}^{-}_{z,k})\boldsymbol{W}^{(r+1:q_k^-)}\right\}\end{aligned}$$

and the second being a Cholesky downdate on the intermediate result, i.e.

$$\boldsymbol{S}^{-}_{zz,k} = \textsc{cholup\_uut}\left\{\boldsymbol{S}^{\dagger}, (\boldsymbol{z}_k^{-(1:r)} \ominus \boldsymbol{m}^{-}_{z,k})\boldsymbol{W}^{(1:r)}, -1\right\}\,,$$

where

$$\boldsymbol{W} = \mathrm{diag}\left\{\sqrt{|w^{-(1)}_{c,k}|}, \ldots, \sqrt{|w^{-(q_k^-)}_{c,k}|}\right\}$$

Then, after computing $\boldsymbol{K}_{x,k}$, the state error square-root factor can be updated via the same sequence of n_z rank-1 Cholesky updates as in the linearization-based case. That is, set $\boldsymbol{S}^* = \boldsymbol{S}^-_{xx,k}$, sequentially perform

$$\boldsymbol{S}^* \leftarrow \text{CHOLUP\_UUT}\left\{\boldsymbol{S}^*, \boldsymbol{u}_i, -1\right\}, \qquad i \in \{1, \ldots, n_z\},$$

and, once all n_z downdates are completed, set $\boldsymbol{S}^+_{xx,k} = \boldsymbol{S}^*$.

The mean update is unaffected by these developments and so the mean update is still given by Eq. (2.6), repeated here as

$$\boldsymbol{m}^+_{x,k} = \boldsymbol{m}^-_{x,k} + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}^-_{z,k}). \tag{2.31}$$

Summary. The quadrature-based square-root filter is summarized in Table 2.5.

Remark 2.9. *The inclusion of the extra Cholesky downdate steps to account for negative quadrature weights is an inelegant but necessary methodology when one is only equipped with the tools of QR-factorization and Cholesky downdating. Section 3.2.4 details new improvements that obviate the need for this two-step approach and instead performs the necessary operations in a single step.*

Table 2.5. Quadrature-based square-root formulation of the MMSE filter.

Models	$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{k-1})$ $\qquad \boldsymbol{S}_{ww,k-1}\boldsymbol{S}_{ww,k-1}^T = \mathbb{E}\left\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\right\}$ $\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k)$ $\qquad \boldsymbol{S}_{vv,k}\boldsymbol{S}_{vv,k}^T = \mathbb{E}\left\{\boldsymbol{v}_k\boldsymbol{v}_k^T\right\}$
Initialization	$\boldsymbol{m}_{x,0} = \mathbb{E}\left\{\boldsymbol{x}_0\right\}$ $\qquad \boldsymbol{S}_{xx,0}\boldsymbol{S}_{xx,0}^T = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})^T\right\}$
Predictor	Draw points: $\boldsymbol{x}_{k-1}^{+(\ell)}$ and $\boldsymbol{w}_{k-1}^{(\ell)}$ with weights $w_{m,k-1}^{+(\ell)}, w_{c,k-1}^{+(\ell)}$ for $\ell \in \{1, \ldots, q_{k-1}^+\}$. $\boldsymbol{x}_k^{-(\ell)} = \boldsymbol{f}(\boldsymbol{x}_{k-1}^{+(\ell)}, \boldsymbol{w}_{k-1}^{(\ell)})$ $\boldsymbol{m}_{x,k}^- = \sum_{\ell=1}^{q_{k-1}^+} w_{m,k-1}^{+(\ell)} \boldsymbol{x}_k^{-(\ell)}$ $\boldsymbol{W} = \operatorname{diag}\left\{\sqrt{\lvert w_{c,k-1}^{+(1)}\rvert}, \ldots, \sqrt{\lvert w_{c,k-1}^{+(q_{k-1}^+)}\rvert}\right\}$ $\boldsymbol{S}^* = \text{HR}\left\{(\boldsymbol{x}_k^{-(r+1:q_{k-1}^+)} \ominus \boldsymbol{m}_{x,k}^-)\boldsymbol{W}^{(r+1:q_{k-1}^+)}\right\}$ $\boldsymbol{S}_{xx,k}^- = \text{CHOLUP\_UUT}\left\{\boldsymbol{S}^*, (\boldsymbol{x}_k^{-(1:r)} \ominus \boldsymbol{m}_{x,k}^-)\boldsymbol{W}^{(1:r)}, -1\right\}$
Corrector	Draw points: $\boldsymbol{x}_k^{-(\ell)}$ and $\boldsymbol{v}_k^{(\ell)}$ with weights $w_{m,k}^{-(\ell)}, w_{c,k}^{-(\ell)}$ for $\ell \in \{1, \ldots, q_k^-\}$. $\boldsymbol{z}_k^{-(\ell)} = \boldsymbol{h}(\boldsymbol{x}_k^{-(\ell)}, \boldsymbol{v}_k^{(\ell)})$ $\boldsymbol{m}_{z,k}^- = \sum_{\ell=1}^{q_k^-} w_{m,k}^{-(\ell)} \boldsymbol{z}_k^{-(\ell)}$ $\boldsymbol{m}_{x,k}^+ = \boldsymbol{m}_{x,k}^- + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^-)$ $\boldsymbol{K}_{x,k} = \boldsymbol{U}_k(\boldsymbol{S}_{zz,k}^-)^{-1}$ $\boldsymbol{U}_k = \boldsymbol{P}_{xz,k}^-(\boldsymbol{S}_{zz,k}^-)^{-T}$ $\boldsymbol{W} = \operatorname{diag}\left\{\sqrt{\lvert w_{c,k}^{-(1)}\rvert}, \ldots, \sqrt{\lvert w_{c,k}^{-(q_k^-)}\rvert}\right\}$ $\boldsymbol{S}^\dagger = \text{HR}\left\{(\boldsymbol{z}_k^{-(r+1:q_k^-)} \ominus \boldsymbol{m}_{z,k}^-)\boldsymbol{W}^{(r+1:q_k^-)}\right\}$ $\boldsymbol{S}_{zz,k}^- = \text{CHOLUP\_UUT}\left\{\boldsymbol{S}^\dagger, (\boldsymbol{z}_k^{-(1:r)} \ominus \boldsymbol{m}_{z,k}^-)\boldsymbol{W}^{(1:r)}, -1\right\}$ $\boldsymbol{P}_{xz,k}^- = \sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)}(\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^-)(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)^T$ Set $\boldsymbol{S}^* = \boldsymbol{S}_{xx,k}^-$ and for $i \in \{1, \ldots, n_z\}$ compute $\boldsymbol{S}^* \leftarrow \text{CHOLUP\_UUT}\left\{\boldsymbol{S}^*, \boldsymbol{u}_i, -1\right\}$ Set $\boldsymbol{S}_{xx,k}^+ = \boldsymbol{S}^*$

3. ADVANCES IN CONSIDER FILTERING

Careful modeling of the deterministic and stochastic motion of the dynamical and observational systems of interest is vital to the performance of any filter. Mismatches between the employed models and the true system, such as systematic errors caused by incorrect model parameters, are a common cause of filter divergence in both real-time and off-line filtering applications. Oftentimes, estimating the values of these parameters is either too computationally burdensome or, due to observability or approximation effects, drastically damaging to filter performance, even causing filter divergence. This divergence happens in a few ways, but the common impacts are (i) accumulated, unaccounted for biases degrading linearization/statistical approximation performance and (ii) over-convergence in the poorly-estimated parameter channels [50]. Regardless, the result is usually the same: loss of the symmetric, positive definite (SPD) characteristic of the error covariance. Choosing, instead, to neglect these errors entirely is a strategy doomed to fail in most cases.

Instead, Stanley F. Schmidt developed a new technique for Apollo and C-5 navigation called "consider filtering" that, in lieu of estimating or neglecting parameter errors, *considers* their impact on the state uncertainty without estimating them directly [16, 18, 55, 56]. Accounting for model mismatch, in the form of appropriate statistical quantification, without estimating the troublesome parameters, can produce a much more stable filter for applications with uncertain models [57, 58]. Additionally, early investigations found that some collections of parameters, while theoretically observable, become linearly dependent due to numerical imprecision, ultimately causing filter divergence [55]. Treating these troublesome parameters as consider parameters greatly reduces the sensitivity to these effects.

There are conceivable cases where even the advantages of consider filtering are compromised numerically. For example, in the case where no or very weak correlations exist between the state and consider parameter vectors, and especially when the parameter uncertainties are small with respect to the state uncertainties, the standard consider filter's equations become numerically very similar to that of a standard Kalman filter. Therefore,

this situation has the potential to succumb to the numerical issues that are common to Kalman filtering [59]. Section 2 identified a number of causes of filter divergence and illustrated how square-root filters can be used to mitigate the effects of these causes, but these filters do not permit the inclusion of consider parameters in their design. Despite the widespread use of square-root filtering and consider filtering for practical applications, a square-root form of the consider filter has proven elusive. This section produces a closed-form recursion for the square-root consider filter.

This section proposes a new filtering technique that possesses the benefits of both square-root *and* consider filters, and the result is an estimation scheme conducive to versatile and numerically stable implementation. Additionally, the new filtering equations are analyzed for ways to produce more efficient implementations. Recent work by Geeraeart [60] has presented an algorithm for a square-root formulation of the consider filter that utilizes the unscented transform (based on the work presented in [52] and [61]); the method in this section was developed independently and proceeds significantly further than the work in [60].

Additionally, there are a number of circumstances where mean and covariance (or square-root factor) are insufficient descriptors of the underlying state vector statistics. In these cases, it is popular to employ nonlinear estimators based upon Bayes' rule, where the state density is characterized as a weighted sum of Gaussians, called a Gaussian mixture (GM). The result is an incredibly powerful and flexible tool, since a GM can arbitrarily approximate any function with a finite number of discontinuities [62]. Despite this flexibility, a GM consider filter is absent from literature and is, therefore, an additional focus of this section.

Section Structure. Similar to the presentation of Section 2, the traditional, full covariance formulation of the consider filter is presented, and those results are then directly leveraged to produce a square-root formulation of the consider filter. Section 3.1 describes MMSE consider filtering, and formulations for both linearization- and quadrature-based approximations to nonlinear system models are presented. Section 3.2 derives the new square-root consider filter using a technique called hyperbolic Householder reflections, and a numerical example is presented to illustrate the advantages and performance of the filter.

Then, Section 3.3 describes how to obtain an analog to MMSE consider filtering using Bayes' rule. The section goes on to derive a more general consider filter that utilizes GM state densities, and the resulting filter is explored in a numerical simulation.

3.1. MINIMUM MEAN SQUARE ERROR CONSIDER FILTERS

Consider the discrete-time, nonlinear dynamical system

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1}) \tag{3.1a}$$

$$\boldsymbol{c}_k = \boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1}) \,, \tag{3.1b}$$

where $\boldsymbol{x}_{k-1}$ represents the state of the system at time t_{k-1}, $\boldsymbol{c}_{k-1}$ represents parameters in the system that are to be considered (but not estimated) at time t_{k-1}, $\boldsymbol{w}_{k-1}$ is zero-mean, white process noise driving the states, $\boldsymbol{u}_{k-1}$ is zero-mean, white process noise driving the consider parameters, $\boldsymbol{f}(\cdot,\cdot,\cdot)$ is the nonlinear dynamics governing the discrete-time evolution of the states, and $\boldsymbol{g}(\cdot,\cdot)$ is the nonlinear dynamics governing the discrete-time evolution of the consider parameters. Accompanying the dynamical system are partial observations of the state, which are governed by the nonlinear measurement model

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) \,, \tag{3.2}$$

where $\boldsymbol{z}_k$ is the measurement at time t_k, $\boldsymbol{v}_k$ is zero-mean, white measurement noise that corrupts the measurement, and $\boldsymbol{h}(\cdot,\cdot,\cdot)$ is the nonlinear measurement function. Where needed, the dimension of the state, consider parameters, and measurement are represented by n_x, n_c, and n_z, respectively. Additionally, the dimensions of the noises $\boldsymbol{w}_{k-1}$, $\boldsymbol{u}_{k-1}$, and $\boldsymbol{v}_k$ are represented by n_w, n_u, and n_v, respectively. Recall that the superscript "+" denotes an *a posteriori* quantity and, the superscript "−" is used to denote an *a priori* quantity.

It is assumed throughout the remainder of this work that the state process noise, consider parameter process noise, and measurement noise are all mutually uncorrelated and that they are all uncorrelated with the state and the consider parameters, themselves.

Furthermore, the initial mean and covariance of the state are given by

$$\begin{aligned}\boldsymbol{m}_{x,0} &= \mathbb{E}\left\{\boldsymbol{x}_0\right\} \\ \boldsymbol{P}_{xx,0} &= \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})^T\right\} .\end{aligned}$$

Similarly, the initial mean and covariance of the consider parameters are given by

$$\begin{aligned}\boldsymbol{m}_{c,0} &= \mathbb{E}\left\{\boldsymbol{c}_0\right\} \\ \boldsymbol{P}_{cc,0} &= \mathbb{E}\left\{(\boldsymbol{c}_0 - \boldsymbol{m}_{c,0})(\boldsymbol{c}_0 - \boldsymbol{m}_{c,0})^T\right\} ,\end{aligned}$$

the cross-covariance between the state and the consider parameters is

$$\boldsymbol{P}_{xc,0} = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{c}_0 - \boldsymbol{m}_{c,0})^T\right\} ,$$

and the covariances of the state process noise, consider parameter process noise, and measurement noise are

$$\begin{aligned}\boldsymbol{P}_{ww,k-1} &= \mathbb{E}\left\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\right\} \\ \boldsymbol{P}_{uu,k-1} &= \mathbb{E}\left\{\boldsymbol{u}_{k-1}\boldsymbol{u}_{k-1}^T\right\} \\ \boldsymbol{P}_{vv,k} &= \mathbb{E}\left\{\boldsymbol{v}_k\boldsymbol{v}_k^T\right\} ,\end{aligned}$$

where it is recalled that the process and measurement noises are all taken to be zero mean and white. Finally, it is noted that each of the initial means and (cross) covariances, as well as the process and measurement noise covariances, are taken to be known.

3.1.1. Predictor. The predicted means of the state and the consider parameters are given by taking the expected value of Eqs. (3.1), yielding

$$\boldsymbol{m}_{x,k}^- = \mathbb{E}\left\{\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1})\right\} \tag{3.3a}$$

$$\boldsymbol{m}_{c,k}^- = \mathbb{E}\left\{\boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1})\right\} . \tag{3.3b}$$

Similarly, the predicted covariances of the state and the consider parameters, as well as the cross-covariance between the state and the consider parameters are found by taking the expected value of the mean-square deviations from the means, such that

$$\boldsymbol{P}^{-}_{xx,k} = \mathbb{E}\left\{(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}^{-}_{x,k})(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}^{-}_{x,k})^T\right\} \tag{3.4a}$$

$$\boldsymbol{P}^{-}_{cc,k} = \mathbb{E}\left\{(\boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1}) - \boldsymbol{m}^{-}_{c,k})(\boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1}) - \boldsymbol{m}^{-}_{c,k})^T\right\} \tag{3.4b}$$

$$\boldsymbol{P}^{-}_{xc,k} = \mathbb{E}\left\{(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}^{-}_{x,k})(\boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1}) - \boldsymbol{m}^{-}_{c,k})^T\right\}. \tag{3.4c}$$

It is worth noting that the same results would be obtained if the state and the consider parameters were formed into an augmented state, expectations were computed, and the result was decomposed into its constituent elements; however, explicitly separating the state and the consider parameters automatically avoids the computation of the cross-covariance between the consider parameters and the state, i.e. $\boldsymbol{P}^{-}_{cx,k}$, since it is simply given by $(\boldsymbol{P}^{-}_{xc,k})^T$.

3.1.2. Corrector. The posterior estimate of the state is constructed as a linear combination of the measurement $\boldsymbol{z}_k$, and the posterior estimate of the consider parameters is taken to be the prior estimate of the consider parameters; that is, the consider parameter estimate is *not* updated. Thus, the *a posteriori* means are

$$\begin{aligned}\boldsymbol{m}^{+}_{x,k} &= \boldsymbol{a}_k + \boldsymbol{K}_{x,k}\boldsymbol{z}_k \\ \boldsymbol{m}^{+}_{c,k} &= \boldsymbol{m}^{-}_{c,k}\,,\end{aligned}$$

where $\boldsymbol{a}_k$ and $\boldsymbol{K}_{x,k}$ are deterministic parameters that are to be determined. Defining the *a posteriori* state estimation error to be $\boldsymbol{e}^{+}_{x,k} = \boldsymbol{x}_k - \boldsymbol{m}^{+}_{x,k}$ and enforcing an unbiased estimator (i.e. $\mathbb{E}\left\{\boldsymbol{e}^{+}_{x,k}\right\} = \boldsymbol{0}_{n_x \times 1}$), it follows that the linear, unbiased estimator is

$$\boldsymbol{m}^{+}_{x,k} = \boldsymbol{m}^{-}_{x,k} + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}^{-}_{z,k}) \tag{3.5a}$$

$$\boldsymbol{m}^{+}_{c,k} = \boldsymbol{m}^{-}_{c,k}\,, \tag{3.5b}$$

where the parameter $\boldsymbol{K}_{x,k}$ is still to be determined and

$$\boldsymbol{m}_{z,k}^{-} = \mathbb{E}\left\{\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k)\right\}$$

is the mean of the nonlinear measurement model. The parameter $\boldsymbol{K}_{x,k}$ is determined such that posterior mean-square state estimation error is minimized, giving rise to the linear, unbiased, minimum mean-square error estimate. From the definition of the posterior state estimation error, it was seen in Section 2 that, for any $\boldsymbol{K}_{x,k}$, the posterior state covariance, $\boldsymbol{P}_{xx,k}^{+} = \mathbb{E}\left\{\boldsymbol{e}_{x,k}^{+}(\boldsymbol{e}_{x,k}^{+})^T\right\}$, is [23]

$$\boldsymbol{P}_{xx,k}^{+} = \boldsymbol{P}_{xx,k}^{-} - \boldsymbol{P}_{xz,k}^{-}\boldsymbol{K}_{x,k}^{T} - \boldsymbol{K}_{x,k}(\boldsymbol{P}_{xz,k}^{-})^T + \boldsymbol{K}_{x,k}\boldsymbol{P}_{zz,k}^{-}\boldsymbol{K}_{x,k}^{T}\,, \tag{3.6}$$

where

$$\begin{aligned}
\boldsymbol{P}_{xz,k}^{-} &= \mathbb{E}\left\{(\boldsymbol{x}_k - \boldsymbol{m}_{x,k}^{-})(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^{-})^T\right\} \\
\boldsymbol{P}_{zz,k}^{-} &= \mathbb{E}\left\{(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^{-})(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^{-})^T\right\}\,.
\end{aligned}$$

The parameter $\boldsymbol{K}_{x,k}$ is found such that

$$J = \mathbb{E}\left\{(\boldsymbol{e}_{x,k}^{+})^T\boldsymbol{e}_{x,k}^{+}\right\} = \mathrm{tr}\{\boldsymbol{P}_{xx,k}^{+}\}$$

is minimized, where $\mathrm{tr}\{\cdot\}$ represents the trace of the input matrix. This optimization problem has the well-known solution

$$\boldsymbol{K}_{x,k} = \boldsymbol{P}_{xz,k}^{-}(\boldsymbol{P}_{zz,k}^{-})^{-1}\,, \tag{3.7}$$

as was seen before. Substituting Eq. (3.7) into Eq. (3.6), it follows that the posterior state covariance may also be expressed as

$$\boldsymbol{P}_{xx,k}^{+} = \boldsymbol{P}_{xx,k}^{-} - \boldsymbol{K}_{x,k}\boldsymbol{P}_{zz,k}^{-}\boldsymbol{K}_{x,k}^{T}\,,$$

which is valid only for the optimal gain given in Eq. (3.7). Since the consider parameter estimate is not updated, as shown by Eq. (3.5b), it follows that the posterior consider parameter covariance is equal to the prior consider parameter covariance, i.e.

$$\boldsymbol{P}_{cc,k}^{+} = \boldsymbol{P}_{cc,k}^{-} \,.$$

While the consider parameter estimate and its covariance are not updated, the cross-covariance between the state and the consider parameter is updated, which is due to the fact that the state is updated. By definition, the cross-covariance

$$\boldsymbol{P}_{xc,k}^{+} = \mathbb{E}\left\{(\boldsymbol{x}_k - \boldsymbol{m}_{x,k}^{+})(\boldsymbol{c}_k - \boldsymbol{m}_{c,k}^{+})^T\right\} \,.$$

It is worth noting that if the consider parameters were to be updated, the corresponding gain would take the familiar form

$$\boldsymbol{K}_{c,k} = \boldsymbol{P}_{cz,k}^{-}(\boldsymbol{P}_{zz,k}^{-})^{-1} \,, \tag{3.8}$$

where the cross covariance term $\boldsymbol{P}_{cz,k}^{-}$ is given by the expectation

$$\boldsymbol{P}_{cz,k}^{-} = \mathbb{E}\left\{(\boldsymbol{c}_k - \boldsymbol{m}_{c,k}^{-})(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^{-})^T\right\} \,.$$

Then, making use of the forms of $\boldsymbol{m}_{x,k}^{+}$ and $\boldsymbol{m}_{c,k}^{+}$ given in Eqs. (3.5), it can be shown that

$$\boldsymbol{P}_{xc,k}^{+} = \boldsymbol{P}_{xc,k}^{-} - \boldsymbol{K}_{x,k}(\boldsymbol{P}_{cz,k}^{-})^T \,,$$

which is valid for any gain $\boldsymbol{K}_{x,k}$.

Summary. The general MMSE consider filter is summarized below and in Table 3.1.

The predicted means, covariances, and cross-covariance for the state and consider parameters are given by

$$
\begin{aligned}
\boldsymbol{m}_{x,k}^{-} &= \mathbb{E}\left\{\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1})\right\} \\
\boldsymbol{m}_{c,k}^{-} &= \mathbb{E}\left\{\boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1})\right\} \\
\boldsymbol{P}_{xx,k}^{-} &= \mathbb{E}\left\{(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}_{x,k}^{-})(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}_{x,k}^{-})^{T}\right\} \\
\boldsymbol{P}_{cc,k}^{-} &= \mathbb{E}\left\{(\boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1}) - \boldsymbol{m}_{c,k}^{-})(\boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1}) - \boldsymbol{m}_{c,k}^{-})^{T}\right\} \\
\boldsymbol{P}_{xc,k}^{-} &= \mathbb{E}\left\{(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}_{x,k}^{-})(\boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1}) - \boldsymbol{m}_{c,k}^{-})^{T}\right\} .
\end{aligned}
$$

The updated means, covariances, and cross-covariance for the state and consider parameters are

$$\boldsymbol{m}_{x,k}^{+} = \boldsymbol{m}_{x,k}^{-} + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^{-}) \tag{3.9a}$$

$$\boldsymbol{m}_{c,k}^{+} = \boldsymbol{m}_{c,k}^{-} \tag{3.9b}$$

$$\boldsymbol{P}_{xx,k}^{+} = \boldsymbol{P}_{xx,k}^{-} - \boldsymbol{K}_{x,k}\boldsymbol{P}_{zz,k}^{-}\boldsymbol{K}_{x,k}^{T} \tag{3.9c}$$

$$\boldsymbol{P}_{cc,k}^{+} = \boldsymbol{P}_{cc,k}^{-} \tag{3.9d}$$

$$\boldsymbol{P}_{xc,k}^{+} = \boldsymbol{P}_{xc,k}^{-} - \boldsymbol{K}_{x,k}(\boldsymbol{P}_{cz,k}^{-})^{T} , \tag{3.9e}$$

where the gain is computed from

$$\boldsymbol{K}_{x,k} = \boldsymbol{P}_{xz,k}^{-}(\boldsymbol{P}_{zz,k}^{-})^{-1} , \tag{3.10}$$

and the required expectations for the update are defined as

$$\boldsymbol{m}_{z,k}^{-} = \mathbb{E}\left\{\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k)\right\} \tag{3.11a}$$

$$\boldsymbol{P}_{zz,k}^{-} = \mathbb{E}\left\{(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^{-})(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^{-})^{T}\right\} \tag{3.11b}$$

$$\boldsymbol{P}_{xz,k}^{-} = \mathbb{E}\left\{(\boldsymbol{x}_k - \boldsymbol{m}_{x,k}^{-})(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^{-})^{T}\right\} \tag{3.11c}$$

$$\boldsymbol{P}_{cz,k}^{-} = \mathbb{E}\left\{(\boldsymbol{c}_k - \boldsymbol{m}_{c,k}^{-})(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^{-})^{T}\right\} . \tag{3.11d}$$

Table 3.1. General formulation of the MMSE consider filter.

Models	$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1})$	$\boldsymbol{P}_{ww,k-1} = \mathbb{E}\left\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\right\}$
	$\boldsymbol{c}_k = \boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1})$	$\boldsymbol{P}_{uu,k-1} = \mathbb{E}\left\{\boldsymbol{u}_{k-1}\boldsymbol{u}_{k-1}^T\right\}$
	$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k)$	$\boldsymbol{P}_{vv,k} = \mathbb{E}\left\{\boldsymbol{v}_k\boldsymbol{v}_k^T\right\}$
Initialization	$\boldsymbol{m}_{x,0} = \mathbb{E}\{\boldsymbol{x}_0\}$	$\boldsymbol{P}_{xx,0} = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})^T\right\}$
	$\boldsymbol{m}_{c,0} = \mathbb{E}\{\boldsymbol{c}_0\}$	$\boldsymbol{P}_{cc,0} = \mathbb{E}\left\{(\boldsymbol{c}_0 - \boldsymbol{m}_{c,0})(\boldsymbol{c}_0 - \boldsymbol{m}_{c,0})^T\right\}$
		$\boldsymbol{P}_{xc,0} = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{c}_0 - \boldsymbol{m}_{c,0})^T\right\}$
Predictor	$\boldsymbol{m}_{x,k}^- = \mathbb{E}\{\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1})\}$	
	$\boldsymbol{m}_{c,k}^- = \mathbb{E}\{\boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1})\}$	
	$\boldsymbol{P}_{xx,k}^- = \mathbb{E}\left\{(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}_{x,k}^-)(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}_{x,k}^-)^T\right\}$	
	$\boldsymbol{P}_{cc,k}^- = \mathbb{E}\left\{(\boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1}) - \boldsymbol{m}_{c,k}^-)(\boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1}) - \boldsymbol{m}_{c,k}^-)^T\right\}$	
	$\boldsymbol{P}_{xc,k}^- = \mathbb{E}\left\{(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1}) - \boldsymbol{m}_{x,k}^-)(\boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1}) - \boldsymbol{m}_{c,k}^-)^T\right\}$	
Corrector	$\boldsymbol{m}_{x,k}^+ = \boldsymbol{m}_{x,k}^- + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^-)$	$\boldsymbol{P}_{xx,k}^+ = \boldsymbol{P}_{xx,k}^- - \boldsymbol{K}_{x,k}\boldsymbol{P}_{zz,k}^-\boldsymbol{K}_{x,k}^T$
	$\boldsymbol{m}_{c,k}^+ = \boldsymbol{m}_{c,k}^-$	$\boldsymbol{P}_{cc,k}^+ = \boldsymbol{P}_{cc,k}^-$
	$\boldsymbol{K}_{x,k} = \boldsymbol{P}_{xz,k}^-(\boldsymbol{P}_{zz,k}^-)^{-1}$	$\boldsymbol{P}_{xc,k}^+ = \boldsymbol{P}_{xc,k}^- - \boldsymbol{K}_{x,k}(\boldsymbol{P}_{cz,k}^-)^T$
	$\boldsymbol{m}_{z,k}^- = \mathbb{E}\{\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k)\}$	
	$\boldsymbol{P}_{zz,k}^- = \mathbb{E}\left\{(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^-)(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^-)^T\right\}$	
	$\boldsymbol{P}_{xz,k}^- = \mathbb{E}\left\{(\boldsymbol{x}_k - \boldsymbol{m}_{x,k}^-)(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^-)^T\right\}$	
	$\boldsymbol{P}_{cz,k}^- = \mathbb{E}\left\{(\boldsymbol{c}_k - \boldsymbol{m}_{c,k}^-)(\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) - \boldsymbol{m}_{z,k}^-)^T\right\}$	

The filtering recursion is initialized at time $t_{k-1} = t_0$ with $\boldsymbol{m}_{x,k-1}^+ = \boldsymbol{m}_{x,0}$, $\boldsymbol{m}_{c,k-1}^+ = \boldsymbol{m}_{c,0}$, $\boldsymbol{P}_{xx,k-1}^+ = \boldsymbol{P}_{xx,0}$, $\boldsymbol{P}_{cc,k-1}^+ = \boldsymbol{P}_{cc,0}$, and $\boldsymbol{P}_{xc,k-1}^+ = \boldsymbol{P}_{xc,0}$.

It is worth remarking that the above equations are agnostic to the method that is used to compute the expectations required for propagating and updating the means, covariances, and cross-covariance; all that is required is that the expectations be computed, and the above recursion can be established. The method employed for determining the expectations therefore dictates the type of consider filter that is implemented.

3.1.3. Linearization-based Approach. For linearization-based formulations of the filter, the nonlinear functions governing the discrete-time evolution of the states and consider parameters, as well as the nonlinear function governing the measurements are all expanded in a first-order Taylor series, thereby neglecting any higher-order effects. If the nonlinear functions involved are actually linear, then no series expansion or truncation is required.

3.1.3.1. Predictor. To establish the expected values described in Section 3.1.1 for the propagation equations, the nonlinear dynamics for the state and the consider parameters are expanded in a first-order Taylor series about the posterior estimates at time t_{k-1}, yielding

$$\begin{aligned}\boldsymbol{f}(\boldsymbol{x}_{k-1},\boldsymbol{c}_{k-1},\boldsymbol{w}_{k-1}) &= \boldsymbol{f}(\boldsymbol{m}^+_{x,k-1},\boldsymbol{m}^+_{c,k-1},\boldsymbol{0}_{n_w\times 1}) + \boldsymbol{F}_{x,k-1}(\boldsymbol{x}_{k-1}-\boldsymbol{m}^+_{x,k-1}) \\ &\quad + \boldsymbol{F}_{c,k-1}(\boldsymbol{c}_{k-1}-\boldsymbol{m}^+_{c,k-1}) + \boldsymbol{F}_{w,k-1}\boldsymbol{w}_{k-1} \\ \boldsymbol{g}(\boldsymbol{c}_{k-1},\boldsymbol{u}_{k-1}) &= \boldsymbol{g}(\boldsymbol{m}^+_{c,k-1},\boldsymbol{0}_{n_u\times 1}) + \boldsymbol{G}_{c,k-1}(\boldsymbol{c}_{k-1}-\boldsymbol{m}^+_{c,k-1}) + \boldsymbol{G}_{u,k-1}\boldsymbol{u}_{k-1}\,,\end{aligned}$$

where the Jacobians may be defined via shorthand notation as

$$\boldsymbol{F}_{\alpha,k-1} = \left[\left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}_{k-1},\boldsymbol{c}_{k-1},\boldsymbol{w}_{k-1})}{\partial \boldsymbol{\alpha}_{k-1}}\right|_{(\cdot)^+_{k-1}}\right] \quad \text{and} \quad \boldsymbol{G}_{\beta,k-1} = \left[\left.\frac{\partial \boldsymbol{g}(\boldsymbol{c}_{k-1},\boldsymbol{u}_{k-1})}{\partial \boldsymbol{\beta}_{k-1}}\right|_{(\cdot)^+_{k-1}}\right].$$

It is to be understood that $\boldsymbol{\alpha}_{k-1}$ can represent $\boldsymbol{x}_{k-1}$, $\boldsymbol{c}_{k-1}$, or $\boldsymbol{w}_{k-1}$, leading to the definitions of $\boldsymbol{F}_{x,k-1}$, $\boldsymbol{F}_{c,k-1}$, and $\boldsymbol{F}_{w,k-1}$, respectively. Additionally, the subscript $(\cdot)^+_{k-1}$ indicates that each Jacobian is evaluated at the posterior means of the state and the consider parameters, where it recalled that the process noises are taken to be zero mean. Similarly, $\boldsymbol{\beta}_{k-1}$ can represent $\boldsymbol{c}_{k-1}$ or $\boldsymbol{u}_{k-1}$, leading to the definitions of $\boldsymbol{G}_{c,k-1}$ and $\boldsymbol{G}_{u,k-1}$, respectively. Applying the first-order Taylor series expansion to the expectations leads to linearized propagation equations of the form

$$\boldsymbol{m}^-_{x,k} = \boldsymbol{f}(\boldsymbol{m}^+_{x,k-1},\boldsymbol{m}^+_{c,k-1},\boldsymbol{0}_{n_w\times 1}) \tag{3.12a}$$

$$\boldsymbol{m}^-_{c,k} = \boldsymbol{g}(\boldsymbol{m}^+_{c,k-1},\boldsymbol{0}_{n_u\times 1}) \tag{3.12b}$$

$$\begin{aligned}\boldsymbol{P}^-_{xx,k} &= \boldsymbol{F}_{x,k-1}\boldsymbol{P}^+_{xx,k-1}\boldsymbol{F}^T_{x,k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{P}^+_{cc,k-1}\boldsymbol{F}^T_{c,k-1} + \boldsymbol{F}_{w,k-1}\boldsymbol{P}_{ww,k-1}\boldsymbol{F}^T_{w,k-1} \\ &\quad + \boldsymbol{F}_{x,k-1}\boldsymbol{P}^+_{xc,k-1}\boldsymbol{F}^T_{c,k-1} + \boldsymbol{F}_{c,k-1}(\boldsymbol{P}^+_{xc,k-1})^T\boldsymbol{F}^T_{x,k-1}\end{aligned} \tag{3.12c}$$

$$\boldsymbol{P}^-_{cc,k} = \boldsymbol{G}_{c,k-1}\boldsymbol{P}^+_{cc,k-1}\boldsymbol{G}^T_{c,k-1} + \boldsymbol{G}_{u,k-1}\boldsymbol{P}_{uu,k-1}\boldsymbol{G}^T_{u,k-1} \tag{3.12d}$$

$$\boldsymbol{P}^-_{xc,k} = \boldsymbol{F}_{x,k-1}\boldsymbol{P}^+_{xc,k-1}\boldsymbol{G}_{c,k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{P}^+_{cc,k-1}\boldsymbol{G}_{c,k-1}\,, \tag{3.12e}$$

where it is assumed that the posterior at t_{k-1} is unbiased.

3.1.3.2. Corrector. Much as with the dynamics, the nonlinear measurement function is expanded in a first-order Taylor series to yield

$$\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) = \boldsymbol{h}(\boldsymbol{m}^-_{x,k}, \boldsymbol{m}^-_{c,k}, \boldsymbol{0}_{n_v\times 1}) + \boldsymbol{H}_{x,k}(\boldsymbol{x}_k - \boldsymbol{m}^-_{x,k}) + \boldsymbol{H}_{c,k}(\boldsymbol{c}_k - \boldsymbol{m}^-_{c,k}) + \boldsymbol{H}_{v,k}\boldsymbol{v}_k\,,$$

where

$$\boldsymbol{H}_{\gamma,k} = \left[\left.\frac{\partial \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k)}{\partial \boldsymbol{\gamma}_k}\right|_{(\cdot)^-_k}\right]$$

represents the general-form Jacobian. In this case, $\boldsymbol{\gamma}_k$ can represent $\boldsymbol{x}_k$, $\boldsymbol{c}_k$, or $\boldsymbol{v}_k$, leading to the definitions of $\boldsymbol{H}_{x,k}$, $\boldsymbol{H}_{c,k}$, and $\boldsymbol{H}_{v,k}$, respectively, and the subscript $(\cdot)^-_k$ indicates that each Jacobian is evaluated at the prior means of the state and the consider parameters, where it recalled that the measurement noise is taken to be zero mean. Applying the first-order Taylor series expansion to the expectations required to compute the update yields

$$\boldsymbol{m}^-_{z,k} = \boldsymbol{h}(\boldsymbol{m}^-_{x,k}, \boldsymbol{m}^-_{c,k}, \boldsymbol{0}_{n_v\times 1}) \tag{3.13a}$$

$$\begin{aligned}\boldsymbol{P}^-_{zz,k} = {} & \boldsymbol{H}_{x,k}\boldsymbol{P}^-_{xx,k}\boldsymbol{H}^T_{x,k} + \boldsymbol{H}_{c,k}\boldsymbol{P}^-_{cc,k}\boldsymbol{H}^T_{c,k} + \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}^T_{v,k} \\ & + \boldsymbol{H}_{x,k}\boldsymbol{P}^-_{xc,k}\boldsymbol{H}^T_{c,k} + \boldsymbol{H}_{c,k}(\boldsymbol{P}^-_{xc,k})^T\boldsymbol{H}^T_{x,k}\end{aligned} \tag{3.13b}$$

$$\boldsymbol{P}^-_{xz,k} = \boldsymbol{P}^-_{xx,k}\boldsymbol{H}^T_{x,k} + \boldsymbol{P}^-_{xc,k}\boldsymbol{H}^T_{c,k} \tag{3.13c}$$

$$\boldsymbol{P}^-_{cz,k} = (\boldsymbol{P}^-_{xc,k})^T\boldsymbol{H}^T_{x,k} + \boldsymbol{P}^-_{cc,k}\boldsymbol{H}^T_{c,k}\,, \tag{3.13d}$$

where it is assumed that the prior at t_k is unbiased. The update is then completed through Eqs. (3.9) and Eq. (3.10).

Summary. The linearization-based MMSE consider filter is summarized in Table 3.2.

3.1.4. Quadrature-based Approach. For quadrature-based formulations of the consider filter, the evaluations of the expectations required are computed via a quadrature scheme, such as Gauss-Hermite quadrature [26], cubature [27], or the unscented transform [28, 29]. While each of the aforementioned methods has differences in how the quadrature points and weights are generated, they may all be treated under the common framework

Table 3.2. Linearization-based formulation of the MMSE consider filter.

Models	$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1})$	$\boldsymbol{P}_{ww,k-1} = \mathbb{E}\left\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\right\}$
	$\boldsymbol{c}_k = \boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1})$	$\boldsymbol{P}_{uu,k-1} = \mathbb{E}\left\{\boldsymbol{u}_{k-1}\boldsymbol{u}_{k-1}^T\right\}$
	$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k)$	$\boldsymbol{P}_{vv,k} = \mathbb{E}\left\{\boldsymbol{v}_k\boldsymbol{v}_k^T\right\}$
Initialization	$\boldsymbol{m}_{x,0} = \mathbb{E}\left\{\boldsymbol{x}_0\right\}$	$\boldsymbol{P}_{xx,0} = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})^T\right\}$
	$\boldsymbol{m}_{c,0} = \mathbb{E}\left\{\boldsymbol{c}_0\right\}$	$\boldsymbol{P}_{cc,0} = \mathbb{E}\left\{(\boldsymbol{c}_0 - \boldsymbol{m}_{c,0})(\boldsymbol{c}_0 - \boldsymbol{m}_{c,0})^T\right\}$
		$\boldsymbol{P}_{xc,0} = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{c}_0 - \boldsymbol{m}_{c,0})^T\right\}$
Predictor	$\boldsymbol{m}_{x,k}^- = \boldsymbol{f}(\boldsymbol{m}_{x,k-1}^+, \boldsymbol{m}_{c,k-1}^+, \boldsymbol{0}_{n_w\times 1})$	
	$\boldsymbol{m}_{c,k}^- = \boldsymbol{g}(\boldsymbol{m}_{c,k-1}^+, \boldsymbol{0}_{n_u\times 1})$	
	$\boldsymbol{P}_{xx,k}^- = \boldsymbol{F}_{x,k-1}\boldsymbol{P}_{xx,k-1}^+\boldsymbol{F}_{x,k-1}^T + \boldsymbol{F}_{c,k-1}\boldsymbol{P}_{cc,k-1}^+\boldsymbol{F}_{c,k-1}^T + \boldsymbol{F}_{w,k-1}\boldsymbol{P}_{ww,k-1}\boldsymbol{F}_{w,k-1}^T$	
	$\quad + \boldsymbol{F}_{x,k-1}\boldsymbol{P}_{xc,k-1}^+\boldsymbol{F}_{c,k-1}^T + \boldsymbol{F}_{c,k-1}(\boldsymbol{P}_{xc,k-1}^+)^T\boldsymbol{F}_{x,k-1}^T$	
	$\boldsymbol{P}_{cc,k}^- = \boldsymbol{G}_{c,k-1}\boldsymbol{P}_{cc,k-1}^+\boldsymbol{G}_{c,k-1}^T + \boldsymbol{G}_{u,k-1}\boldsymbol{P}_{uu,k-1}\boldsymbol{G}_{u,k-1}^T$	
	$\boldsymbol{P}_{xc,k}^- = \boldsymbol{F}_{x,k-1}\boldsymbol{P}_{xc,k-1}^+\boldsymbol{G}_{c,k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{P}_{cc,k-1}^+\boldsymbol{G}_{c,k-1}$	
Corrector	$\boldsymbol{m}_{x,k}^+ = \boldsymbol{m}_{x,k}^- + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^-)$	$\boldsymbol{P}_{xx,k}^+ = \boldsymbol{P}_{xx,k}^- - \boldsymbol{K}_{x,k}\boldsymbol{P}_{zz,k}^-\boldsymbol{K}_{x,k}^T$
	$\boldsymbol{m}_{c,k}^+ = \boldsymbol{m}_{c,k}^-$	$\boldsymbol{P}_{cc,k}^+ = \boldsymbol{P}_{cc,k}^-$
	$\boldsymbol{K}_{x,k} = \boldsymbol{P}_{xz,k}^-(\boldsymbol{P}_{zz,k}^-)^{-1}$	$\boldsymbol{P}_{xc,k}^+ = \boldsymbol{P}_{xc,k}^- - \boldsymbol{K}_{x,k}(\boldsymbol{P}_{cz,k}^-)^T$
	$\boldsymbol{m}_{z,k}^- = \boldsymbol{h}(\boldsymbol{m}_{x,k}^-, \boldsymbol{m}_{c,k}^-, \boldsymbol{0}_{n_v\times 1})$	
	$\boldsymbol{P}_{zz,k}^- = \boldsymbol{H}_{x,k}\boldsymbol{P}_{xx,k}^-\boldsymbol{H}_{x,k}^T + \boldsymbol{H}_{c,k}\boldsymbol{P}_{cc,k}^-\boldsymbol{H}_{c,k}^T + \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}_{v,k}^T$	
	$\quad + \boldsymbol{H}_{x,k}\boldsymbol{P}_{xc,k}^-\boldsymbol{H}_{c,k}^T + \boldsymbol{H}_{c,k}(\boldsymbol{P}_{xc,k}^-)^T\boldsymbol{H}_{x,k}^T$	
	$\boldsymbol{P}_{xz,k}^- = \boldsymbol{P}_{xx,k}^-\boldsymbol{H}_{x,k}^T + \boldsymbol{P}_{xc,k}^-\boldsymbol{H}_{c,k}^T$	
	$\boldsymbol{P}_{cz,k}^- = (\boldsymbol{P}_{xc,k}^-)^T\boldsymbol{H}_{x,k}^T + \boldsymbol{P}_{cc,k}^-\boldsymbol{H}_{c,k}^T$	

of quadrature integration. Given the nonlinear transformation

$$\boldsymbol{\alpha} = \boldsymbol{\varphi}(\boldsymbol{\beta})\,,$$

where $\boldsymbol{\beta}$ has a known mean, $\boldsymbol{m}_\beta$, and covariance $\boldsymbol{P}_{\beta\beta}$, the mean and covariance of $\boldsymbol{\alpha}$, along with the cross-covariance between $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are approximated by

$$\boldsymbol{m}_\alpha = \textstyle\sum_{\ell=1}^q w_m^{(\ell)}\boldsymbol{\varphi}(\boldsymbol{\beta}^{(\ell)}) \tag{3.14a}$$

$$\boldsymbol{P}_{\alpha\alpha} = \textstyle\sum_{\ell=1}^q w_c^{(\ell)}(\boldsymbol{\varphi}(\boldsymbol{\beta}^{(\ell)}) - \boldsymbol{m}_\alpha)(\boldsymbol{\varphi}(\boldsymbol{\beta}^{(\ell)}) - \boldsymbol{m}_\alpha)^T \tag{3.14b}$$

$$\boldsymbol{P}_{\beta\alpha} = \textstyle\sum_{\ell=1}^q w_c^{(\ell)}(\boldsymbol{\beta}^{(\ell)} - \boldsymbol{m}_\beta)(\boldsymbol{\varphi}(\boldsymbol{\beta}^{(\ell)}) - \boldsymbol{m}_\alpha)^T\,, \tag{3.14c}$$

where $\boldsymbol{\beta}^{(\ell)}$ are the quadrature points, $w_m^{(\ell)}$ and $w_c^{(\ell)}$ are the quadrature weights for the mean and covariance, respectively, and q is the number of quadrature points.

The difference between Gauss-Hermite quadrature, cubature, and the unscented transform is in how the quadrature points and weights are chosen. As the focus of the present work is on consider filters, only some salient aspects of the quadrature schemes are discussed. Gauss-Hermite quadrature typically relies on choosing the same quadrature points for each of Eqs. (3.14). As a consequence, the Gauss-Hermite quadrature weights are the same for the mean and covariance transformations, i.e. $w_m^{(\ell)} = w_c^{(\ell)} \;\; \forall \; \ell \in \{1, 2, \ldots, q\}$. The cubature approach also employs the same quadrature weights in the mean and covariance transformations. The unscented transform, however, in its most general form, selects different quadrature weights for the mean and covariance transformations in Eqs. (3.14). One thing that is specific to the unscented transform approach is that a negative weight can be assigned to one of the quadrature points, which has significant implications on positivity in the covariance transformations [53].

When the nonlinear function under consideration has multiple inputs, i.e. when it takes the form

$$\boldsymbol{\alpha} = \boldsymbol{\varphi}(\boldsymbol{\beta}, \boldsymbol{\gamma}) \,,$$

quadrature techniques can still be leveraged. In this case, the inputs are concatenated to form a single input $(\boldsymbol{\beta}')^T = [\boldsymbol{\beta}^T \quad \boldsymbol{\gamma}^T]$. The mean and the covariance of the composite input are determined, including the cross-covariance between $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, and the previously described approach to determining the mean, covariance, and cross-covariance (with all inputs) of $\boldsymbol{\alpha}$ is utilized. When one, or more, of the inputs appear linearly, analytic integration can be, at least partially, used to transform the mean and covariance. This, however, is a specific case that is easily handled, so it will not be considered moving forward. If more than two inputs are required for a given transformation, the same approach of concatenation is applied.

3.1.4.1. Predictor. For the propagation stage of the consider filter, the expected values described in Section 3.1.1 must be determined. As both of the dynamic equations representing the evolution of the state and the consider parameters, given by Eqs. (3.1), contain multiple inputs, concatenation of the inputs is required. Moreover, Eqs. (3.1) possess

common inputs. To account for this dependency, the augmented mean and covariance at time t_{k-1} are formed as

$$\breve{\boldsymbol{m}}^{+}_{k-1} = \begin{bmatrix} \boldsymbol{m}^{+}_{x,k-1} \\ \boldsymbol{m}^{+}_{c,k-1} \\ \boldsymbol{0}_{n_w\times 1} \\ \boldsymbol{0}_{n_u\times 1} \end{bmatrix} \qquad \text{and} \qquad \breve{\boldsymbol{P}}^{+}_{k-1} = \begin{bmatrix} \boldsymbol{P}^{+}_{xx,k-1} & \boldsymbol{P}^{+}_{xc,k-1} & \boldsymbol{0}_{n_x\times n_w} & \boldsymbol{0}_{n_x\times n_u} \\ (\boldsymbol{P}^{+}_{xc,k-1})^T & \boldsymbol{P}^{+}_{cc,k-1} & \boldsymbol{0}_{n_c\times n_w} & \boldsymbol{0}_{n_c\times n_u} \\ \boldsymbol{0}_{n_w\times n_x} & \boldsymbol{0}_{n_w\times n_c} & \boldsymbol{P}_{ww,k-1} & \boldsymbol{0}_{n_w\times n_u} \\ \boldsymbol{0}_{n_u\times n_x} & \boldsymbol{0}_{n_u\times n_c} & \boldsymbol{0}_{n_u\times n_w} & \boldsymbol{P}_{uu,k-1} \end{bmatrix},$$

where it is recalled that the state and consider parameter process noises are taken to be zero mean and are assumed to be mutually uncorrelated and uncorrelated with the state and consider parameters. Leveraging the concatenated mean and covariance, a set of q^{+}_{k-1} quadrature points and weights of the form $\breve{\boldsymbol{x}}^{+(\ell)}_{k-1}$, $w^{+(\ell)}_{m,k-1}$, and $w^{+(\ell)}_{c,k-1}$ are generated. The quadrature points are then partitioned into state, consider parameter, state process noise, and consider parameter process noise quadrature points as

$$(\breve{\boldsymbol{x}}^{+(\ell)}_{k-1})^T = [(\boldsymbol{x}^{+(\ell)}_{k-1})^T \quad (\boldsymbol{c}^{+(\ell)}_{k-1})^T \quad (\boldsymbol{w}^{(\ell)}_{k-1})^T \quad (\boldsymbol{u}^{(\ell)}_{k-1})^T]\,. \tag{3.15}$$

Using the partitioned quadrature points, a set of transformed quadrature points is formed by subjecting the input quadrature points to Eqs. (3.1), which yields

$$\begin{aligned} \boldsymbol{x}^{-(\ell)}_{k} &= \boldsymbol{f}(\boldsymbol{x}^{+(\ell)}_{k-1}, \boldsymbol{c}^{+(\ell)}_{k-1}, \boldsymbol{w}^{(\ell)}_{k-1}) \\ \boldsymbol{c}^{-(\ell)}_{k} &= \boldsymbol{g}(\boldsymbol{c}^{+(\ell)}_{k-1}, \boldsymbol{u}^{(\ell)}_{k-1})\,, \end{aligned}$$

and the expected values describing the propagation step of the consider filter are given by

$$\boldsymbol{m}^{-}_{x,k} = \textstyle\sum_{\ell=1}^{q^{+}_{k-1}} w^{+(\ell)}_{m,k-1} \boldsymbol{x}^{-(\ell)}_{k} \tag{3.16a}$$

$$\boldsymbol{m}^{-}_{c,k} = \textstyle\sum_{\ell=1}^{q^{+}_{k-1}} w^{+(\ell)}_{m,k-1} \boldsymbol{c}^{-(\ell)}_{k} \tag{3.16b}$$

$$\boldsymbol{P}^{-}_{xx,k} = \textstyle\sum_{\ell=1}^{q^{+}_{k-1}} w^{+(\ell)}_{c,k-1} (\boldsymbol{x}^{-(\ell)}_{k} - \boldsymbol{m}^{-}_{x,k})(\boldsymbol{x}^{-(\ell)}_{k} - \boldsymbol{m}^{-}_{x,k})^T \tag{3.16c}$$

$$\boldsymbol{P}^{-}_{cc,k} = \textstyle\sum_{\ell=1}^{q^{+}_{k-1}} w^{+(\ell)}_{c,k-1} (\boldsymbol{c}^{-(\ell)}_{k} - \boldsymbol{m}^{-}_{c,k})(\boldsymbol{c}^{-(\ell)}_{k} - \boldsymbol{m}^{-}_{c,k})^T \tag{3.16d}$$

$$\boldsymbol{P}^{-}_{xc,k} = \textstyle\sum_{\ell=1}^{q^{+}_{k-1}} w^{+(\ell)}_{c,k-1} (\boldsymbol{x}^{-(\ell)}_{k} - \boldsymbol{m}^{-}_{x,k})(\boldsymbol{c}^{-(\ell)}_{k} - \boldsymbol{m}^{-}_{c,k})^T\,. \tag{3.16e}$$

It is worth noting that the number of quadrature points is denoted by q_{k-1}^{+}. This is to account for the general case that the quadrature rule can be changed through time. Whereas the cubature and unscented transform quadrature methods will always employ the same number of quadrature points for a given input dimension, Gauss-Hermite quadrature can be adapted online to use a varying number of points. This same idea applies to the mean and covariance weights – the parameters defining the weights can be adapted online, most notably with the unscented transform, which can lead to the case where the weights are not constant through time.

3.1.4.2. Corrector. Following the same pattern as described for the propagation stage, an augmented mean and covariance are defined from the *a priori* means, covariances, and cross-covariance of the state and consider parameters, while also including the measurement noise. This leads to the augmented mean and covariance as

$$\breve{\boldsymbol{m}}_k^- = \begin{bmatrix} \boldsymbol{m}_{x,k}^- \\ \boldsymbol{m}_{c,k}^- \\ \boldsymbol{0}_{n_v \times 1} \end{bmatrix} \qquad \text{and} \qquad \breve{\boldsymbol{P}}_k^- = \begin{bmatrix} \boldsymbol{P}_{xx,k}^- & \boldsymbol{P}_{xc,k}^- & \boldsymbol{0}_{n_x \times n_v} \\ (\boldsymbol{P}_{xc,k}^-)^T & \boldsymbol{P}_{cc,k}^- & \boldsymbol{0}_{n_c \times n_v} \\ \boldsymbol{0}_{n_v \times n_x} & \boldsymbol{0}_{n_v \times n_c} & \boldsymbol{P}_{vv,k} \end{bmatrix},$$

where the measurement noise is taken to be zero mean and uncorrelated with the state and consider parameters. A set of q_k^- quadrature points and weights of the form $\breve{\boldsymbol{x}}_k^{-(\ell)}$, $w_{m,k}^{-(\ell)}$, and $w_{c,k}^{-(\ell)}$ is generated and then partitioned according to

$$(\breve{\boldsymbol{x}}_k^{-(\ell)})^T = [(\boldsymbol{x}_k^{-(\ell)})^T \quad (\boldsymbol{c}_k^{-(\ell)})^T \quad (\boldsymbol{v}_k^{(\ell)})^T] \,.$$

The *a priori* quadrature points are transformed through the nonlinear measurement function given by Eq. (3.2) to produce measurement quadrature points of the form

$$\boldsymbol{z}_k^{-(\ell)} = \boldsymbol{h}(\boldsymbol{x}_k^{-(\ell)}, \boldsymbol{c}_k^{-(\ell)}, \boldsymbol{v}_k^{(\ell)}) \,.$$

The expectations necessary for the update are computed as

$$\boldsymbol{m}_{z,k}^{-} = \sum_{\ell=1}^{q_k^-} w_{m,k}^{-(\ell)} \boldsymbol{z}_k^{-(\ell)} \tag{3.17a}$$

$$\boldsymbol{P}_{zz,k}^{-} = \sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)} (\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^{-})(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^{-})^T \tag{3.17b}$$

$$\boldsymbol{P}_{xz,k}^{-} = \sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)} (\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^{-})(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^{-})^T \tag{3.17c}$$

$$\boldsymbol{P}_{cz,k}^{-} = \sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)} (\boldsymbol{c}_k^{-(\ell)} - \boldsymbol{m}_{c,k}^{-})(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^{-})^T \,, \tag{3.17d}$$

and the update is completed with Eqs. (3.9) and Eq. (3.10). As with the propagation, the number of quadrature points in the update, q_k^-, is potentially allowed to vary through time. More importantly, the number of quadrature points used in the propagation stage is not at all required to be the same as the number of quadrature points used in the update. As discussed with the propagation stage, the mean and covariance weights can also change over time, specifically if the parameters of the unscented transform are allowed to change in time, leading to the inclusion of a time index on the weights.

Summary. The quadrature-based formulation of the MMSE consider filter is summarized in Table 3.3.

3.2. SQUARE-ROOT CONSIDER FILTERS

This section derives the square-root consider filter, but first, the cornerstone of the derivation must be discussed: hyperbolic Householder reflections. It was seen in Section 2 that the keys to the standard square-root filter were RQ-factorization (accomplished using Householder reflections) and Cholesky downdating. Here, it is demonstrated that hyperbolic Householder reflections, in some sense, marry these two concepts into a single numerical procedure. Then, the predictor/corrector relationships for the linearization- and quadrature-based square-root consider filters are developed. Afterward, a collection of improvements to standard (i.e. non-consider) square-root filters are presented as a corollary to the preceding developments, and a method for drastically improving the computational

Table 3.3. Quadrature-based formulation of the MMSE consider filter.

Models	$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1})$	$\boldsymbol{P}_{ww,k-1} = \mathbb{E}\left\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\right\}$
	$\boldsymbol{c}_k = \boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1})$	$\boldsymbol{P}_{uu,k-1} = \mathbb{E}\left\{\boldsymbol{u}_{k-1}\boldsymbol{u}_{k-1}^T\right\}$
	$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k)$	$\boldsymbol{P}_{vv,k} = \mathbb{E}\left\{\boldsymbol{v}_k\boldsymbol{v}_k^T\right\}$
Initialization	$\boldsymbol{m}_{x,0} = \mathbb{E}\left\{\boldsymbol{x}_0\right\}$	$\boldsymbol{P}_{xx,0} = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})^T\right\}$
	$\boldsymbol{m}_{c,0} = \mathbb{E}\left\{\boldsymbol{c}_0\right\}$	$\boldsymbol{P}_{cc,0} = \mathbb{E}\left\{(\boldsymbol{c}_0 - \boldsymbol{m}_{c,0})(\boldsymbol{c}_0 - \boldsymbol{m}_{c,0})^T\right\}$
		$\boldsymbol{P}_{xc,0} = \mathbb{E}\left\{(\boldsymbol{x}_0 - \boldsymbol{m}_{x,0})(\boldsymbol{c}_0 - \boldsymbol{m}_{c,0})^T\right\}$
	Draw points: $\boldsymbol{x}_{k-1}^{+(\ell)},\ \boldsymbol{c}_{k-1}^{+(\ell)},\ \boldsymbol{w}_{k-1}^{(\ell)},$ and $\boldsymbol{u}_{k-1}^{(\ell)}$ with weights $w_{m,k-1}^{+(\ell)}, w_{c,k-1}^{+(\ell)}$ for $\ell \in \{1, \ldots, q_{k-1}^+\}$.	
Predictor	$\boldsymbol{x}_k^{-(\ell)} = \boldsymbol{f}(\boldsymbol{x}_{k-1}^{+(\ell)}, \boldsymbol{c}_{k-1}^{+(\ell)}, \boldsymbol{w}_{k-1}^{(\ell)})$	$\boldsymbol{c}_k^{-(\ell)} = \boldsymbol{g}(\boldsymbol{c}_{k-1}^{+(\ell)}, \boldsymbol{u}_{k-1}^{(\ell)})$
	$\boldsymbol{m}_{x,k}^- = \sum_{\ell=1}^{q_{k-1}^+} w_{m,k-1}^{+(\ell)} \boldsymbol{x}_k^{-(\ell)}$	$\boldsymbol{P}_{xx,k}^- = \sum_{\ell=1}^{q_{k-1}^+} w_{c,k-1}^{+(\ell)} (\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^-)(\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^-)^T$
	$\boldsymbol{m}_{c,k}^- = \sum_{\ell=1}^{q_{k-1}^+} w_{m,k-1}^{+(\ell)} \boldsymbol{c}_k^{-(\ell)}$	$\boldsymbol{P}_{cc,k}^- = \sum_{\ell=1}^{q_{k-1}^+} w_{c,k-1}^{+(\ell)} (\boldsymbol{c}_k^{-(\ell)} - \boldsymbol{m}_{c,k}^-)(\boldsymbol{c}_k^{-(\ell)} - \boldsymbol{m}_{c,k}^-)^T$
		$\boldsymbol{P}_{xc,k}^- = \sum_{\ell=1}^{q_{k-1}^+} w_{c,k-1}^{+(\ell)} (\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^-)(\boldsymbol{c}_k^{-(\ell)} - \boldsymbol{m}_{c,k}^-)^T$
	Draw points: $\boldsymbol{x}_k^{-(\ell)},\ \boldsymbol{c}_k^{-(\ell)},$ and $\boldsymbol{v}_k^{(\ell)}$ with weights $w_{m,k}^{-(\ell)}, w_{c,k}^{-(\ell)}$ for $\ell \in \{1, \ldots, q_k^-\}$.	
Corrector	$\boldsymbol{m}_{x,k}^+ = \boldsymbol{m}_{x,k}^- + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^-)$	$\boldsymbol{P}_{xx,k}^+ = \boldsymbol{P}_{xx,k}^- - \boldsymbol{K}_{x,k}\boldsymbol{P}_{zz,k}^-\boldsymbol{K}_{x,k}^T$
	$\boldsymbol{m}_{c,k}^+ = \boldsymbol{m}_{c,k}^-$	$\boldsymbol{P}_{cc,k}^+ = \boldsymbol{P}_{cc,k}^-$
	$\boldsymbol{K}_{x,k} = \boldsymbol{P}_{xz,k}^-(\boldsymbol{P}_{zz,k}^-)^{-1}$	$\boldsymbol{P}_{xc,k}^+ = \boldsymbol{P}_{xc,k}^- - \boldsymbol{K}_{x,k}(\boldsymbol{P}_{cz,k}^-)^T$
	$\boldsymbol{z}_k^{-(\ell)} = \boldsymbol{h}(\boldsymbol{x}_k^{-(\ell)}, \boldsymbol{c}_k^{-(\ell)}, \boldsymbol{v}_k^{(\ell)})$	
	$\boldsymbol{m}_{z,k}^- = \sum_{\ell=1}^{q_k^-} w_{m,k}^{-(\ell)} \boldsymbol{z}_k^{-(\ell)}$	
	$\boldsymbol{P}_{zz,k}^- = \sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)} (\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)^T$	
	$\boldsymbol{P}_{xz,k}^- = \sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)} (\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^-)(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)^T$	
	$\boldsymbol{P}_{cz,k}^- = \sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)} (\boldsymbol{c}_k^{-(\ell)} - \boldsymbol{m}_{c,k}^-)(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)^T$	

efficiency of the new square-root consider filter's time updates for certain systems is presented. Finally, the section concludes with a numerical simulation to evaluate the filter's performance and its advantages over standard approaches.

3.2.1. Hyperbolic Householder Reflections. In 1986, Rader generalized the Householder method of reflections for matrix triangularization to solve data addition and deletion problems for least squares [63]. Inspired by the work of Bunse-Gerstner in [64], he derived a method for Householder reflections that can preserve negative signs. That is, whereas traditional Householder reflections can be used to triangularize the result of matrix addition, such as obtaining a triangular $\boldsymbol{S}$ as the result of

$$\boldsymbol{S}\boldsymbol{S}^T = \boldsymbol{X}\boldsymbol{X}^T + \boldsymbol{Y}\boldsymbol{Y}^T,$$

Rader's hyperbolic Householder reflections can be used to triangularize addition or subtraction, such as

$$\boldsymbol{S}\boldsymbol{S}^T = \boldsymbol{X}\boldsymbol{X}^T \pm \boldsymbol{Y}\boldsymbol{Y}^T .$$

In fact, hyperbolic Householder reflections can simultaneously account for addition *and* subtraction, such as problems like

$$\boldsymbol{S}\boldsymbol{S}^T = \boldsymbol{X}\boldsymbol{X}^T - \boldsymbol{Y}\boldsymbol{Y}^T + \boldsymbol{Z}\boldsymbol{Z}^T .$$

Despite the effectiveness of this new class of reflections, and despite the thorough analysis and compelling results presented by Rader, this technique has largely fallen into obscurity. It is hoped that this dissertation's discussion on the hyperbolic Householder reflection, the types of problems it can solve, and the new filter formulations derived using it in this section will offer new insights for solving even more problems in navigation, estimation, and beyond.

The implications of hyperbolic Householder reflections are enormous, as they, in some sense, marry the RQ-decomposition (or equivalently QR-decomposition), the Cholesky update, and the Cholesky downdate. As will be demonstrated, each of these problems can be treated using hyperbolic Householder reflections, but first some details about the technique are presented.

The hyperbolic Householder reflection relies on the concept of *hypernormality*; that is, any orthogonal matrix $\boldsymbol{\Xi}$ that satisfies

$$\boldsymbol{\Xi}\boldsymbol{Y}\boldsymbol{\Xi}^T = \boldsymbol{Y} \tag{3.18}$$

is said to be hypernormal with respect to the matrix $\boldsymbol{Y}$, which is diagonal with entries of ± 1. An important aspect of hypernormality is that simply providing $\boldsymbol{\Xi}$ and indicating it is a hypernormal matrix is an incomplete description. One must also indicate the matrix $\boldsymbol{Y}$ with which $\boldsymbol{\Xi}$ is hypernormal to, and accordingly, $\boldsymbol{Y}$ will herein be referred to as the "signature matrix." The matrix $\boldsymbol{\Xi}$ is always guaranteed to be nonsingular, and, while nonsymmetric

in general, always obeys so-called hyperbolic symmetry such that

$$\boldsymbol{\Xi Y \Xi}^T = \boldsymbol{\Xi}^T \boldsymbol{Y \Xi} \,.$$

Proof of these properties is presented in Appendix C.1.

The terminology "hypernormal" is inspired by the fact that a hypernormal matrix preserves the hyperbolic norm of a vector. This is to say, if

$$\boldsymbol{u}^T \boldsymbol{Y u} = \sum_i |u_i|^2 Y_{i,i}$$

denotes the hyperbolic norm, called as such because hyperbolic functions are often in the form of sums and differences of squares, then, if $\boldsymbol{v} = \boldsymbol{\Xi}^T \boldsymbol{u}$,

$$\boldsymbol{v}^T \boldsymbol{Y v} = \boldsymbol{u}^T \boldsymbol{Y u} \,. \tag{3.19}$$

Proof of Eq. (3.19) is given in Appendix C.2.

The hyperbolic Householder reflection is produced by defining the matrix, denoted $\boldsymbol{\Xi}$ to distinguish it from the Householder reflection matrix $\boldsymbol{Q}$ of Section 2.3.1.2,

$$\boldsymbol{\Xi} = \boldsymbol{Y} - \frac{2\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T \boldsymbol{Y v}} \,,$$

for some vector $\boldsymbol{v}$, where it can immediately be noted that if $\boldsymbol{Y} = \boldsymbol{I}$, the hyperbolic Householder reflection becomes the traditional Householder reflection. Unlike the traditional reflections, however, the hyperbolic Householder reflection can embed negative ones in $\boldsymbol{Y}$ to perform subtractions (and, similarly, can embed positive ones to simultaneously perform additions, too). Just like the traditional counterpart, a sequence of these reflections can be used to transform a rectangular matrix into an upper triangular matrix, that is

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \end{bmatrix} \bar{\boldsymbol{\Xi}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{U} \end{bmatrix} \,,$$

where $\bar{\boldsymbol{\Xi}}$ denotes the full sequence of reflections and $\boldsymbol{U}$ is an upper triangular matrix. In contrast to the traditional reflections, problems that involve addition *and* subtraction can be considered. In particular, this technique will be used to find upper triangular factors $\tilde{\boldsymbol{S}}$ resulting from "up-and-downdate" problems of the form

$$\tilde{\boldsymbol{S}}\tilde{\boldsymbol{S}}^T = \boldsymbol{S}\boldsymbol{S}^T - \boldsymbol{A}\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{B}^T \,.$$

In Remark 2.6 on pp. 44, the relationship between the RQ-decomposition and the Cholesky update was illustrated. In similar fashion, focus now turns to demonstrating that the RQ-decomposition, the Cholesky update, and the Cholesky downdate are subsumed by the capabilities of hyperbolic Householder reflections. This is to say that in the tasks performed by these three methods can instead be performed by this single methodology:

1. Traditional Householder reflection-based RQ-decompositions can be performed with hyperbolic Householder reflections
2. Cholesky updates can be performed with hyperbolic Householder reflections
3. Cholesky downdates can be performed with hyperbolic Householder reflections

Performing traditional Householder RQ-decompositions with hyperbolic Householder reflections is trivially done by simply performing the hyperbolic rotations according to $\boldsymbol{Y} = \boldsymbol{I}$, and, furthermore, the relationship between the RQ-decomposition and the Cholesky update was already illustrated in the aforementioned remark. Two of the three claims have thus been supported, and what remains is demonstrate the third and final claim. To do so, recall that the Cholesky downdate is of the form

$$\tilde{\boldsymbol{S}}\tilde{\boldsymbol{S}}^T = \boldsymbol{S}\boldsymbol{S} - \boldsymbol{u}\boldsymbol{u}^T \,.$$

Rather than solving this problem using a rank-1 Cholesky downdate, instead factor this expression as

$$\tilde{\boldsymbol{S}}\tilde{\boldsymbol{S}}^T = \begin{bmatrix} \boldsymbol{u} & \boldsymbol{S} \end{bmatrix} \boldsymbol{Y} \begin{bmatrix} \boldsymbol{u}^T \\ \boldsymbol{S}^T \end{bmatrix} ,$$

where

$$\boldsymbol{Y} = \text{blkdiag}\{-1, \boldsymbol{I}_{\dim\{\boldsymbol{S}\}}\},$$

blkdiag$\{\cdot\}$ is used to denote block diagonal concatenation, and $\dim\{\boldsymbol{A}\}$ denotes the dimension of $\boldsymbol{A}$. Now, selecting a sequence of hyperbolic Householder reflections such that $\bar{\boldsymbol{\Xi}}\boldsymbol{Y}\bar{\boldsymbol{\Xi}}^T = \boldsymbol{Y}$ and

$$\begin{bmatrix}\boldsymbol{u} & \boldsymbol{S}\end{bmatrix}\bar{\boldsymbol{\Xi}} = \begin{bmatrix}\boldsymbol{0}_{\dim\{\boldsymbol{u}\}} & \boldsymbol{S}^\dagger\end{bmatrix},$$

i.e. they produce an upper triangular matrix $\boldsymbol{S}^\dagger$, permits the conclusion that

$$\tilde{\boldsymbol{S}} = \boldsymbol{S}^\dagger$$

since

$$\begin{aligned}\tilde{\boldsymbol{S}}\tilde{\boldsymbol{S}}^T &= \begin{bmatrix}\boldsymbol{u} & \boldsymbol{S}\end{bmatrix}\boldsymbol{Y}\begin{bmatrix}\boldsymbol{u}^T \\ \boldsymbol{S}^T\end{bmatrix} \\ &= \begin{bmatrix}\boldsymbol{u} & \boldsymbol{S}\end{bmatrix}\bar{\boldsymbol{\Xi}}\boldsymbol{Y}\bar{\boldsymbol{\Xi}}^T\begin{bmatrix}\boldsymbol{u}^T \\ \boldsymbol{S}^T\end{bmatrix} \\ &= \begin{bmatrix}\boldsymbol{0}_{\dim\{\boldsymbol{u}\}} & \boldsymbol{S}^\dagger\end{bmatrix}\boldsymbol{Y}\begin{bmatrix}\boldsymbol{0}^T_{\dim\{\boldsymbol{u}\}} \\ (\boldsymbol{S}^\dagger)^T\end{bmatrix} \\ &= \boldsymbol{S}^\dagger(\boldsymbol{S}^\dagger)^T.\end{aligned}$$

Therefore, rank-1 Cholesky downdates can be accomplished using hyperbolic Householder reflections. In fact, higher-rank modifications (where $\boldsymbol{u}$ is some matrix $\boldsymbol{U}$) can be performed with these reflections as well, and this will be described in Section 3.2.4.

Moving forward, $\bar{\boldsymbol{\Xi}}$ will simply be written as $\boldsymbol{\Xi}$ since there is no risk of confusing a single reflection with an entire sequence of reflections.

Algorithm 4 provides a procedure to compute the upper triangular square-root factor of the up-and-downdate

$$\boldsymbol{S}\boldsymbol{S}^T - \boldsymbol{A}\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{B}^T \,. \tag{3.20}$$

This is a modification of Rader's algorithm presented in [63] to produce an upper triangular factor. Note that Algorithm 4 is only for problems of exactly of the form of Eq. (3.20) and will produce incorrect results if, for example, the addition and subtraction are interchanged. A user is warned to exhibit caution to ensure that the argument order, $\boldsymbol{S}$, $\boldsymbol{A}$, and $\boldsymbol{B}$, reflect problems of the form in Eq. (3.20).

Algorithm 4 Hyperbolic Householder Reflections for Up-and-Downdating $\boldsymbol{S}$

function HHR_UP_AND_DOWNDATE($\boldsymbol{S}$, $\boldsymbol{A}$, $\boldsymbol{B}$)
  $\boldsymbol{S}$ is $m \times m$, $\boldsymbol{A}$ is $m \times p$, and $\boldsymbol{B}$ is $m \times q$
  $\boldsymbol{X} = [\ \boldsymbol{B} \mid \boldsymbol{A} \mid \boldsymbol{S}\ ]$
  $\boldsymbol{X}$ is $m \times n$, let $x_{i,j}$ denote the $(i,j)^{\text{th}}$ entry of $\boldsymbol{X}$
  $\boldsymbol{Y} = \text{blkdiag}(\boldsymbol{I}_{q\times q}, -\boldsymbol{I}_{p\times p}, \boldsymbol{I}_{(n-p-q)\times(n-p-q)})$
  for $k = 1, \ldots, m$ **do**
    $i = m - k + 1$
    $j = n - k + 1$
    $\boldsymbol{U} = \text{zeros}(n, 1)$
    $\boldsymbol{U}_{1:j} \leftarrow \boldsymbol{X}^T_{i,1:j}$
    $\boldsymbol{E}_j = [0, \ldots, 0, 1, 0, \ldots, 0]^T$ (only a 1 in the j^{th} entry)
    if $|x_{i,j}| = 0$ **then**
      $\sigma = \sqrt{\boldsymbol{U}^T\boldsymbol{Y}\boldsymbol{U}}$
    else
      $\sigma = (x_{i,j}/|x_{i,j}|)\sqrt{\boldsymbol{U}^T\boldsymbol{Y}\boldsymbol{U}}$
    $\boldsymbol{B} = \boldsymbol{Y}\boldsymbol{U} + \sigma\boldsymbol{E}_j$
    $\boldsymbol{Q} = \boldsymbol{Y} - 2\boldsymbol{B}\boldsymbol{B}^T/(\boldsymbol{B}^T\boldsymbol{Y}\boldsymbol{B})$
    $\boldsymbol{X} \leftarrow \boldsymbol{X}\boldsymbol{Q}$
return $\tilde{\boldsymbol{S}} = \boldsymbol{X}_{1:m,n-m+1:n}$

3.2.2. Linearization-based Approach. In the following, the linearization-based covariance propagation and update expressions from Section 3.1.3 are manipulated to produce a square-root formulation of the (linearization-based) consider filter. This section builds off of Section 3.1.3, as this section immediately leverages and manipulates Section 3.1.3's results for covariance propagation and update. As the mean propagation and update equations are algorithmically unaffected by these modifications, and since these

equations are presented in a previous section, discussion of these quantities is omitted. As before, the procedure is assumed to be initialized with a collection of appropriate means and square-root factors of covariance.

3.2.2.1. Predictor. Define a concatenated covariance matrix $\bar{\boldsymbol{P}}_{k-1}^{+}$ as

$$\bar{\boldsymbol{P}}_{k-1}^{+} = \begin{bmatrix} \boldsymbol{P}_{xx,k-1}^{+} & \boldsymbol{P}_{xc,k-1}^{+} \\ \boldsymbol{P}_{cx,k-1}^{+} & \boldsymbol{P}_{cc,k-1}^{+} \end{bmatrix},$$

where $\boldsymbol{P}_{cx,k-1}^{+} = (\boldsymbol{P}_{xc,k-1}^{+})^{T}$. Note that this definition applies similarly for $\bar{\boldsymbol{P}}_{k}^{-}$ and $\bar{\boldsymbol{P}}_{k}^{+}$, but with subscripts and superscripts replaced appropriately in their constituent elements. Then, defining the concatenated system matrices

$$\begin{aligned} \bar{\boldsymbol{F}}_{k-1} &= \begin{bmatrix} \boldsymbol{F}_{x,k-1} & \boldsymbol{F}_{c,k-1} \\ \boldsymbol{0}_{n_c \times n_x} & \boldsymbol{I}_{n_c \times n_c} \end{bmatrix} \\ \bar{\boldsymbol{G}}_{k-1} &= \begin{bmatrix} \boldsymbol{I}_{n_x \times n_x} & \boldsymbol{0}_{n_x \times n_c} \\ \boldsymbol{0}_{n_c \times n_x} & \boldsymbol{G}_{c,k-1} \end{bmatrix} \\ \bar{\boldsymbol{M}}_{k-1} &= \begin{bmatrix} \boldsymbol{F}_{w,k-1} & \boldsymbol{0}_{n_x \times n_c} \\ \boldsymbol{0}_{n_c \times n_x} & \boldsymbol{G}_{u,k-1} \end{bmatrix} \\ \bar{\boldsymbol{Q}}_{k-1} &= \begin{bmatrix} \boldsymbol{P}_{ww,k-1} & \boldsymbol{0}_{n_x \times n_c} \\ \boldsymbol{0}_{n_c \times n_x} & \boldsymbol{P}_{uu,k-1} \end{bmatrix}, \end{aligned}$$

with $\boldsymbol{F}_{\alpha,k-1}$ and $\boldsymbol{G}_{\beta,k-1}$ defined as in Section 3.1.3.1, allows Eqs. (3.12c)–(3.12e) to be written as the single expression

$$\bar{\boldsymbol{P}}_{k}^{-} = \bar{\boldsymbol{G}}_{k-1}\bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{P}}_{k-1}^{+}\bar{\boldsymbol{F}}_{k-1}^{T}\bar{\boldsymbol{G}}_{k-1}^{T} + \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{Q}}_{k-1}\bar{\boldsymbol{M}}_{k-1}^{T}. \tag{3.21}$$

This result is an equivalent but concatenated, and thus notationally compact, representation of Eqs. (3.12c)–(3.12e), and it is this expression that will be manipulated to yield a recursion in terms of square-root factors.

To that end, substituting the concatenated matrix square-root factors

$$\bar{\boldsymbol{P}}_k^- = \bar{\boldsymbol{S}}_k^-(\bar{\boldsymbol{S}}_k^-)^T$$
$$\bar{\boldsymbol{P}}_{k-1}^+ = \bar{\boldsymbol{S}}_{k-1}^+(\bar{\boldsymbol{S}}_{k-1}^+)^T$$
$$\bar{\boldsymbol{Q}}_{k-1} = \bar{\boldsymbol{V}}_{k-1}\bar{\boldsymbol{V}}_{k-1}^T$$

into Eq. (3.21) yields

$$\bar{\boldsymbol{S}}_k^-(\bar{\boldsymbol{S}}_k^-)^T = \bar{\boldsymbol{G}}_{k-1}\bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{S}}_{k-1}^+(\bar{\boldsymbol{S}}_{k-1}^+)^T\bar{\boldsymbol{F}}_{k-1}^T\bar{\boldsymbol{G}}_{k-1}^T + \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{V}}_{k-1}\bar{\boldsymbol{V}}_{k-1}^T\bar{\boldsymbol{M}}_{k-1}^T\,,$$

an expression that can be factored as

$$\bar{\boldsymbol{S}}_k^-(\bar{\boldsymbol{S}}_k^-)^T = \left[\begin{array}{c|c} \bar{\boldsymbol{G}}_{k-1}\bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{S}}_{k-1}^+ & \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{V}}_{k-1} \end{array}\right]\left[\begin{array}{c|c} \bar{\boldsymbol{G}}_{k-1}\bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{S}}_{k-1}^+ & \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{V}}_{k-1} \end{array}\right]^T.$$

This allows the conclusion that a valid *a priori* square-root factor is given as

$$\left[\begin{array}{c|c} \bar{\boldsymbol{G}}_{k-1}\bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{S}}_{k-1}^+ & \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{V}}_{k-1} \end{array}\right],$$

but, as this matrix is non-square, further manipulation is required to obtain the desired result. It holds true that a suitable RQ-factorization that produces an upper triangular matrix of dimension $\bar{n}\times\bar{n}$, with $\bar{n} = n_x + n_c$, yields the desired concatenated *a priori* square-root factor. That is,

$$\bar{\boldsymbol{S}}_k^- = \mathrm{rq}\left\{\left[\begin{array}{c|c} \bar{\boldsymbol{G}}_{k-1}\bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{S}}_{k-1}^+ & \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{V}}_{k-1} \end{array}\right]\right\}.$$

The RQ-decomposition prescribed here can be accomplished via a sequence of Householder reflections using Algorithm 2, such that the final result is obtained as

$$\bar{\boldsymbol{S}}_k^- = \mathrm{HR}\left\{\left[\begin{array}{c|c} \bar{\boldsymbol{G}}_{k-1}\bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{S}}_{k-1}^+ & \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{V}}_{k-1} \end{array}\right]\right\}. \tag{3.22}$$

Therefore, given the *a posteriori* concatenated square-root factor at t_{k-1}, $\bar{\boldsymbol{S}}_{k-1}^+$, the *a priori* concatenated square-root factor at t_k, $\bar{\boldsymbol{S}}_k^-$, can be computed using Eq. (3.22).

While the discussion presented in this section produces a complete and general propagation scheme for the square-root consider filter that employs linearization, Section 3.2.5 illustrates a number of improvements for practical applications that can greatly decrease the expected computational cost of the procedure. In particular, Section 3.2.5 utilizes specific selections of the consider parameters' models to substantially reduce the required computational effort.

3.2.2.2. Corrector. To obtain a square-root form of the linearization-based update equations presented in Section 3.1.3.2, start by defining the square-root factor of the residual covariance as

$$\boldsymbol{P}^{-}_{zz,k} = \boldsymbol{S}^{-}_{zz,k}(\boldsymbol{S}^{-}_{zz,k})^{T},$$

where, from Eq. (3.13b), it can be shown that[1]

$$\boldsymbol{S}^{-}_{zz,k} = \mathrm{rq}\left\{\left[\begin{array}{c|c|c} \boldsymbol{H}_{x,k}\boldsymbol{S}^{-}_{xx,k} & \boldsymbol{H}_{x,k}\boldsymbol{S}^{-}_{xc,k} + \boldsymbol{H}_{c,k}\boldsymbol{S}^{-}_{cc,k} & \boldsymbol{H}_{v,k}\boldsymbol{S}_{vv,k} \end{array}\right]\right\}$$

if

$$\bar{\boldsymbol{S}}^{-}_{k} = \begin{bmatrix} \boldsymbol{S}^{-}_{xx,k} & \boldsymbol{S}^{-}_{xc,k} \\ \boldsymbol{0}_{n_c \times n_x} & \boldsymbol{S}^{-}_{cc,k} \end{bmatrix}.$$

Now, the gains $\boldsymbol{K}_{x,k}$ and $\boldsymbol{K}_{c,k}$, of Eqs. (3.7) and (3.8), respectively, are expressed in terms of square-root factors as

$$\boldsymbol{K}_{x,k} = \left[\boldsymbol{P}^{-}_{xz,k}(\boldsymbol{S}^{-}_{zz,k})^{-T}\right](\boldsymbol{S}^{-}_{zz,k})^{-1} \tag{3.23a}$$

$$\boldsymbol{K}_{c,k} = \left[\boldsymbol{P}^{-}_{cz,k}(\boldsymbol{S}^{-}_{zz,k})^{-T}\right](\boldsymbol{S}^{-}_{zz,k})^{-1}, \tag{3.23b}$$

where $(\cdot)^{-T}$ denotes the inverse transposed. While at first these nested inverses may appear troublesome, remembering that $\boldsymbol{S}^{-}_{zz,k}$ is upper triangular indicates that no inversion is required, and that, instead, simple backward substitution efficiently produces the desired

[1] As with Eq. (3.22), this RQ-factorization can be accomplished using the HR$\{\cdot\}$ routine given in Algorithm 2.

matrices.[2] The cross covariance terms $\boldsymbol{P}^-_{xz,k}$ and $\boldsymbol{P}^-_{cz,k}$ are computed using Eqs. (3.13c) and (3.13d) but expressed in terms of square-root factors as

$$\begin{aligned}\boldsymbol{P}^-_{xz,k} &= [\boldsymbol{S}^-_{xx,k}(\boldsymbol{S}^-_{xx,k})^T + \boldsymbol{S}^-_{xc,k}(\boldsymbol{S}^-_{xc,k})^T]\boldsymbol{H}^T_{x,k} + \boldsymbol{S}^-_{xc,k}(\boldsymbol{S}^-_{cc,k})^T\boldsymbol{H}^T_{c,k} \\ \boldsymbol{P}^-_{cz,k} &= \boldsymbol{S}^-_{cc,k}(\boldsymbol{S}^-_{xc,k})^T\boldsymbol{H}^T_{x,k} + \boldsymbol{S}^-_{cc,k}(\boldsymbol{S}^-_{cc,k})^T\boldsymbol{H}^T_{c,k}\,.\end{aligned}$$

Defining the concatenated update gain

$$\bar{\boldsymbol{K}}_k = \begin{bmatrix}\boldsymbol{K}_{x,k} \\ \boldsymbol{K}_{c,k}\end{bmatrix}$$

permits the consider covariance update of Eqs. (3.9c)–(3.9e) can be rewritten in concatenated form as

$$\bar{\boldsymbol{P}}^+_k = \bar{\boldsymbol{P}}^-_k - \bar{\boldsymbol{K}}_k\boldsymbol{P}^-_{zz,k}\bar{\boldsymbol{K}}^T_k + \boldsymbol{\Upsilon}\boldsymbol{K}_{c,k}\boldsymbol{P}^-_{zz,k}\boldsymbol{K}^T_{c,k}\boldsymbol{\Upsilon}^T\,, \tag{3.24}$$

where

$$\boldsymbol{\Upsilon} = \begin{bmatrix}\boldsymbol{0}_{n_x\times n_c} \\ \boldsymbol{I}_{n_c\times n_c}\end{bmatrix}.$$

Again, much like was seen with Eq. (3.21), the equations Eqs. (3.9c)–(3.9e) have been collected into an equivalent, concatenated representation given by Eq. (3.24).

Then, substituting matrix factors such that $\bar{\boldsymbol{P}}^-_k = \bar{\boldsymbol{S}}^-_k(\bar{\boldsymbol{S}}^-_k)^T$, $\bar{\boldsymbol{P}}^+_k = \bar{\boldsymbol{S}}^+_k(\bar{\boldsymbol{S}}^+_k)^T$, and $\boldsymbol{P}^-_{zz,k} = \boldsymbol{S}^-_{zz,k}(\boldsymbol{S}^-_{zz,k})^T$ allows Eq. (3.24) to be written in terms of square-root factors as

$$\bar{\boldsymbol{S}}^+_k(\bar{\boldsymbol{S}}^+_k)^T = \bar{\boldsymbol{S}}^-_k(\bar{\boldsymbol{S}}^-_k)^T - \bar{\boldsymbol{K}}_k\boldsymbol{S}^-_{zz,k}(\boldsymbol{S}^-_{zz,k})^T\bar{\boldsymbol{K}}^T_k + \boldsymbol{\Upsilon}\boldsymbol{K}_{c,k}\boldsymbol{S}^-_{zz,k}(\boldsymbol{S}^-_{zz,k})^T\boldsymbol{K}^T_{c,k}\boldsymbol{\Upsilon}^T\,. \tag{3.25}$$

Note that this can be identified as a rank-n_z *downdate* of $\bar{\boldsymbol{S}}^-_k$ according to $\bar{\boldsymbol{K}}_k\boldsymbol{S}^-_{zz,k}$, followed by a rank-$n_z$ *update* of $\bar{\boldsymbol{S}}^-_k$ according to $\boldsymbol{\Upsilon}\boldsymbol{K}_{c,k}\boldsymbol{S}^-_{zz,k}$. This would typically be implemented as a sequence of n_z rank-1 Cholesky downdates followed by a sequence of n_z rank-1 Cholesky updates. The sequence of downdates performs the full square-root factor update on the

[2] In fact, what is sought is the solution to least squares problem $\boldsymbol{K}_{c,k}\boldsymbol{S}^-_{zz,k}(\boldsymbol{S}^-_{zz,k})^T = \boldsymbol{P}^-_{cz,k}$, a very famous problem structure that is easily solved when $\boldsymbol{S}^-_{zz,k}$ is triangular.

entire system, and the sequence of updates undoes any modifications to the consider parameter square-root factor. While this downdate-then-update procedure will produce the desired square-root factor $\bar{\boldsymbol{S}}_k^+$, a more unified, mathematically appealing, and, perhaps, efficient approach is desired.

By defining the signature matrix

$$\boldsymbol{Y} = \text{blkdiag}\left\{\boldsymbol{I}_{n_z \times n_z}, -\boldsymbol{I}_{n_z \times n_z}, \boldsymbol{I}_{\bar{n} \times \bar{n}}\right\}, \tag{3.26}$$

Eq. (3.25) can be factored as

$$\bar{\boldsymbol{S}}_k^+ (\bar{\boldsymbol{S}}_k^+)^T = \left[\begin{array}{c|c|c} \boldsymbol{\Upsilon}\boldsymbol{K}_{c,k}\boldsymbol{S}_{zz,k}^- & \bar{\boldsymbol{K}}_k\boldsymbol{S}_{zz,k}^- & \bar{\boldsymbol{S}}_k^- \end{array}\right] \boldsymbol{Y} \left[\begin{array}{c|c|c} \boldsymbol{\Upsilon}\boldsymbol{K}_{c,k}\boldsymbol{S}_{zz,k}^- & \bar{\boldsymbol{K}}_k\boldsymbol{S}_{zz,k}^- & \bar{\boldsymbol{S}}_k^- \end{array}\right]^T.$$

Now, recall the matrix $\boldsymbol{\Xi}$ that is hypernormal with respect to $\boldsymbol{Y}$; that is,

$$\boldsymbol{\Xi}\boldsymbol{Y}\boldsymbol{\Xi}^T = \boldsymbol{Y}.$$

This property inspires the designation of $\boldsymbol{Y}$ as a "signature matrix," as $\boldsymbol{\Xi}$ is hypernormal with respect to $\boldsymbol{Y}$ only and not an arbitrary matrix. Then, if $\boldsymbol{\Xi}$ is *also* a matrix such that

$$\left[\begin{array}{c|c|c} \boldsymbol{\Upsilon}\boldsymbol{K}_{c,k}\boldsymbol{S}_{zz,k}^- & \bar{\boldsymbol{K}}_k\boldsymbol{S}_{zz,k}^- & \bar{\boldsymbol{S}}_k^- \end{array}\right] \boldsymbol{\Xi} = \left[\begin{array}{c|c|c} \boldsymbol{0}_{n_x \times n_z} & \boldsymbol{0}_{n_x \times n_z} & \bar{\boldsymbol{S}}_k^\dagger \end{array}\right],$$

it can be concluded that

$$\bar{\boldsymbol{S}}_k^+ = \bar{\boldsymbol{S}}_k^\dagger.$$

Consequently, such a matrix can be computed using a sequence of *hyperbolic* Householder reflections (see Section 3.2.1), and the posterior concatenated square-root factor can be obtained via

$$\bar{\boldsymbol{S}}_k^+ = \text{hhr}\left\{\left[\begin{array}{c|c|c} \boldsymbol{\Upsilon}\boldsymbol{K}_{c,k}\boldsymbol{S}_{zz,k}^- & \bar{\boldsymbol{K}}_k\boldsymbol{S}_{zz,k}^- & \bar{\boldsymbol{S}}_k^- \end{array}\right]\right\},$$

where hhr$\{\cdot\}$ denotes a sequence of hyperbolic Householder reflections to *appropriately* triangularize the matrix argument. The word "appropriately" is emphasized here because the reader is warned not to employ traditional Householder reflections as they do not preserve negative signs (e.g. the $-\boldsymbol{I}_{n_z \times n_z}$ sub-block of $\boldsymbol{Y}$). By contrast, hyperbolic Householder reflections preserve the negative signs in $\boldsymbol{Y}$ and, therefore, perform the factorized update appropriately. A procedure to perform this sequence of hyperbolic Householder reflections for the presented signature matrix $\boldsymbol{Y}$ is outlined as Algorithm 4. Therefore, the posterior square-root factor of the system can be computed using Algorithm 4 as

$$\bar{\boldsymbol{S}}_k^+ = \text{HHR\_UP\_AND\_DOWNDATE}\left\{\bar{\boldsymbol{S}}_k^-,\ \bar{\boldsymbol{K}}_k \boldsymbol{S}_{zz,k}^-,\ \boldsymbol{\Upsilon}\boldsymbol{K}_{c,k}\boldsymbol{S}_{zz,k}^-\right\}. \tag{3.27}$$

The reader should use caution when applying hyperbolic Householder reflections for different problems and formulations than the ones presented here, as each sequence of reflections $\boldsymbol{\Xi}$ is specific to a given signature matrix $\boldsymbol{Y}$. Deviations in the signature matrix from what is described here, i.e. the procedure given in Algorithm 4, will require modifications to produce an appropriate sequence of reflections and final result.

This result is an improvement of the Cholesky up-and-downdate method previously mentioned because it produces the desired, updated square-root factor in one pass, as opposed to performing an update and "undoing" part of it.

Summary. The linearization-based, square-root formulation of the MMSE consider filter is summarized in Table 3.4. Given some $\bar{\boldsymbol{S}}_{k-1}^+$, the consider square-root factor prediction is performed using Eq. (3.22) and the consider square-root factor correction is computed using Eq. (3.27) with gains defined according to Eqs. (3.23). This is repeated for subsequent predictor/corrector cycles as more data become available.

3.2.3. Quadrature-based Approach. The covariance propagation and update expressions from Section 3.1.4 are manipulated to produce a square-root formulation of the quadrature-based consider filter. As with the linearization-based square-root discussion, treatment of the mean propagation and update equations is omitted due to their earlier presentation, and the procedure is assumed to be initialized with appropriate means and square-root factors of covariance. In what follows, the employed quadrature scheme must

Table 3.4. Linearization-based, square-root formulation of the MMSE consider filter.

Models	$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1})$	$\boldsymbol{P}_{ww,k-1} = \boldsymbol{S}_{ww,k-1}\boldsymbol{S}_{ww,k-1}^T = \mathbb{E}\left\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\right\}$
	$\boldsymbol{c}_k = \boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1})$	$\boldsymbol{P}_{uu,k-1} = \boldsymbol{S}_{uu,k-1}\boldsymbol{S}_{uu,k-1}^T = \mathbb{E}\left\{\boldsymbol{u}_{k-1}\boldsymbol{u}_{k-1}^T\right\}$
	$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k)$	$\boldsymbol{P}_{vv,k} = \boldsymbol{S}_{vv,k}\boldsymbol{S}_{vv,k}^T = \mathbb{E}\left\{\boldsymbol{v}_k\boldsymbol{v}_k^T\right\}$
Initialization	$\boldsymbol{m}_{x,0} = \mathbb{E}\left\{\boldsymbol{x}_0\right\}$	$\boldsymbol{P}_{xx,0} = \boldsymbol{S}_{xx,0}\boldsymbol{S}_{xx,0}^T + \boldsymbol{S}_{xc,0}\boldsymbol{S}_{xc,0}^T$
	$\boldsymbol{m}_{c,0} = \mathbb{E}\left\{\boldsymbol{c}_0\right\}$	$\boldsymbol{P}_{cc,0} = \boldsymbol{S}_{cc,0}\boldsymbol{S}_{cc,0}^T$
Predictor	$\boldsymbol{m}_{x,k}^- = \boldsymbol{f}(\boldsymbol{m}_{x,k-1}^+, \boldsymbol{m}_{c,k-1}^+, \boldsymbol{0}_{n_w\times 1})$	
	$\boldsymbol{m}_{c,k}^- = \boldsymbol{g}(\boldsymbol{m}_{c,k-1}^+, \boldsymbol{0}_{n_u\times 1})$	$\bar{\boldsymbol{S}}_{k-1}^+ = \begin{bmatrix} \boldsymbol{S}_{xx,k-1}^+ & \boldsymbol{S}_{xc,k-1}^+ \\ (\boldsymbol{S}_{xc,k-1}^+)^T & \boldsymbol{S}_{cc,k-1}^+ \end{bmatrix}$
	$\bar{\boldsymbol{S}}_k^- = \text{HR}\left\{\left[\begin{array}{c|c} \bar{\boldsymbol{G}}_{k-1}\bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{S}}_{k-1}^+ & \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{V}}_{k-1} \end{array}\right]\right\}$	
Corrector	$\boldsymbol{m}_{z,k}^- = \boldsymbol{h}(\boldsymbol{m}_{x,k}^-, \boldsymbol{m}_{c,k}^-, \boldsymbol{0}_{n_v\times 1})$	
	$\boldsymbol{m}_{x,k}^+ = \boldsymbol{m}_{x,k}^- + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^-)$	$\boldsymbol{m}_{c,k}^+ = \boldsymbol{m}_{c,k}^-$
	$\boldsymbol{K}_{x,k} = \left[\boldsymbol{P}_{xz,k}^-(\boldsymbol{S}_{zz,k}^-)^{-T}\right](\boldsymbol{S}_{zz,k}^-)^{-1}$	$\boldsymbol{K}_{c,k} = \left[\boldsymbol{P}_{cz,k}^-(\boldsymbol{S}_{zz,k}^-)^{-T}\right](\boldsymbol{S}_{zz,k}^-)^{-1}$
	$\bar{\boldsymbol{K}}_k = \begin{bmatrix} \boldsymbol{K}_{x,k} \\ \boldsymbol{K}_{c,k} \end{bmatrix}$	$\boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{0}_{n_x\times n_c} \\ \boldsymbol{I}_{n_c\times n_c} \end{bmatrix}$
	$\bar{\boldsymbol{S}}_k^+ = \text{HHR\_UP\_AND\_DOWNDATE}\left\{\bar{\boldsymbol{S}}_k^-,\ \bar{\boldsymbol{K}}_k\boldsymbol{S}_{zz,k}^-,\ \boldsymbol{\Upsilon}\boldsymbol{K}_{c,k}\boldsymbol{S}_{zz,k}^-\right\}$	
	$\boldsymbol{S}_{zz,k}^- = \text{HR}\left\{\left[\begin{array}{c|c} \boldsymbol{H}_{x,k}\boldsymbol{S}_{xx,k}^- & \boldsymbol{H}_{x,k}\boldsymbol{S}_{xc,k}^- + \boldsymbol{H}_{c,k}\boldsymbol{S}_{cc,k}^- \end{array}\right.\right.$	$\left.\left.\begin{array}{c} \\ \boldsymbol{H}_{v,k}\boldsymbol{S}_{vv,k} \end{array}\right]\right\}$
	$\boldsymbol{P}_{xz,k}^- = [\boldsymbol{S}_{xx,k}^-(\boldsymbol{S}_{xx,k}^-)^T + \boldsymbol{S}_{xc,k}^-(\boldsymbol{S}_{xc,k}^-)^T]\boldsymbol{H}_{x,k}^T + \boldsymbol{S}_{xc,k}^-(\boldsymbol{S}_{cc,k}^-)^T\boldsymbol{H}_{c,k}^T$	
	$\boldsymbol{P}_{cz,k}^- = \boldsymbol{S}_{cc,k}^-(\boldsymbol{S}_{xc,k}^-)^T\boldsymbol{H}_{x,k}^T + \boldsymbol{S}_{cc,k}^-(\boldsymbol{S}_{cc,k}^-)^T\boldsymbol{H}_{c,k}^T$	

be designed to employ square-root factors in lieu of covariances when generating quadrature points. In most quadrature schemes, this simplifies the point selection, as it avoids computing the Cholesky factorization of the concatenated covariance matrix (such as $\breve{\boldsymbol{P}}_{k-1}^+$ in Section 3.1.4.1). Instead, the square-root factor matrix required to generate these points is simply a block concatenation of the square-root factors already stored by the filter.

3.2.3.1. Predictor. Using the definition of concatenated covariances and square-root factors as in Section 3.2.2 (e.g. $\bar{\boldsymbol{P}}_{k-1}^+$, $\bar{\boldsymbol{S}}_{k-1}^+$, etc.), the following details a square-root formulation of the quadrature-based propagation presented in Section 3.2.2.1.

First, subject the points $\breve{\boldsymbol{x}}_{k-1}^{+(\ell)}$ of Eq. (3.15) to Eqs. (3.1) to obtain the predicted set of points $\bar{\boldsymbol{x}}_k^{-(\ell)}$, and rewrite Equations (3.16c)–(3.16e) in the concatenated form

$$\bar{\boldsymbol{P}}_k^- = \sum_{\ell=1}^{q_{k-1}^+} w_{c,k-1}^{+(\ell)}(\bar{\boldsymbol{x}}_k^{-(\ell)} - \bar{\boldsymbol{m}}_k^-)(\bar{\boldsymbol{x}}_k^{-(\ell)} - \bar{\boldsymbol{m}}_k^-)^T\,,$$

where $(\bar{\boldsymbol{x}}_k^{-(\ell)})^T = [(\boldsymbol{x}_k^{-(\ell)})^T \ \ (\boldsymbol{c}_k^{-(\ell)})^T] \in \mathbb{R}^{\bar{n}}$ and $\bar{\boldsymbol{m}}_k^- = \mathbb{E}\left\{\bar{\boldsymbol{x}}_k^{-(\ell)}\right\}$. Additionally, let weights

$$w_{c,k-1}^{+(1)}, w_{c,k-1}^{+(2)}, \ldots, w_{c,k-1}^{+(r)}$$

denote the r weights that have negative values, and let

$$w_{c,k-1}^{+(r+1)}, w_{c,k-1}^{+(r+2)}, \ldots, w_{c,k-1}^{+(q_{k-1}^{+})}$$

denote the remaining weights that have non-negative values. Note that this requires the indexing of the weights to be such that the first r weights have negative values and the remainder are positive. In the case where the employed quadrature rule does not permit negative weights (such as with Gauss-Hermite quadrature), $r = 0$ and the same definitions hold. Similarly, let $\bar{\boldsymbol{x}}_k^{-(i:j)}$ denote a column-wise concatenation of the the i^{th} to j^{th} quadrature points, such that $\bar{\boldsymbol{x}}_k^{-(i:j)}$ is $\bar{n} \times (j - i + 1)$.[3]

Using methods similar to those presented in Section 3.2.2, it can then be shown that

$$\bar{\boldsymbol{S}}_k^{-} = \text{hhr}\left\{ \left[\begin{array}{c|c} \left(\bar{\boldsymbol{x}}_k^{-(1:r)} \ominus \bar{\boldsymbol{m}}_k^{-}\right)\boldsymbol{W}^{(1:r)} & \left(\bar{\boldsymbol{x}}_k^{-(r+1:q_{k-1}^{+})} \ominus \bar{\boldsymbol{m}}_k^{-}\right)\boldsymbol{W}^{(r+1:q_{k-1}^{+})} \end{array} \right] \right\}, \tag{3.28}$$

where

$$\boldsymbol{W} = \text{diag}\left\{ \sqrt{|w_{c,k-1}^{+(1)}|}, \ldots, \sqrt{|w_{c,k-1}^{+(q_{k-1}^{+})}|} \right\},$$

$\boldsymbol{W}^{(i:j)}$ denotes the (square) submatrix of $\boldsymbol{W}$ containing the i^{th} to j^{th} rows and columns of $\boldsymbol{W}$, the hyperbolic Householder reflections are taken with respect to the signature matrix

$$\boldsymbol{Y} = \text{blkdiag}\left\{ -\boldsymbol{I}_{r\times r}, \boldsymbol{I}_{(q_{k-1}^{+}-r)\times(q_{k-1}^{+}-r)} \right\}, \tag{3.29}$$

and $|\cdot|$ denotes the absolute value. Proof is given in Appendix C.3.

With respect to Algorithm 4, the propagated square-root factor can be computed via

$$\bar{\boldsymbol{S}}_k^{-} = \text{HHR\_UP\_AND\_DOWNDATE}\left\{\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{0}_{\bar{n}\times 1}\right\}, \tag{3.30}$$

[3]This work employs the convention that in the case of $r = 0$, $\bar{\boldsymbol{x}}_k^{-(1:r)}$, $\boldsymbol{W}^{(1:r)}$, $\boldsymbol{I}_{r\times r}$, etc. are empty.

where

$$\boldsymbol{U}_1 = (\bar{\boldsymbol{x}}_k^{-(r+1:q_{k-1}^+)} \ominus \bar{\boldsymbol{m}}_k^-)\boldsymbol{W}^{(r+1:q_{k-1}^+)}$$
$$\boldsymbol{U}_2 = \begin{cases} (\bar{\boldsymbol{x}}_k^{-(1:r)} \ominus \bar{\boldsymbol{m}}_k^-)\boldsymbol{W}^{(1:r)} & \text{if } r > 0 \\ \boldsymbol{0}_{\bar{n}\times 1} & \text{otherwise} \end{cases} .$$

The value of this approach is that, in the case of no negative weights ($r = 0$), this prediction reduces precisely to traditional Householder reflections (hyperbolic Householder reflections with identity signature matrix) but is able to *also* treat the $r > 0$ case with no consideration of an extra step that accounts for the negative weights.

It is noted that a more efficient implementation can be devised to avoid the necessary inclusion of $\boldsymbol{0}_{\bar{n}\times 1}$ as the third argument of HHR_UP_AND_DOWNDATE$\{\cdot\}$ via simple modification of the provided algorithm, but such an additional algorithm is omitted for compactness. The need for this extra argument is due to the fact that Algorithm 4 was devised for signature matrices of the form of Eq. (3.26), and this quadrature-based time update is with respect to the signature matrix in Eq. (3.29).

3.2.3.2. Corrector. Given measurement-transformed quadrature points $\boldsymbol{z}_k^{-(\ell)} = \boldsymbol{h}(\boldsymbol{x}_k^{-(\ell)}, \boldsymbol{c}_k^{-(\ell)}, \boldsymbol{v}_k^{(\ell)})$ and the expected measurement $\boldsymbol{m}_{z,k}^-$ given by Eq. (3.17a), the square-root factor $\boldsymbol{S}_{zz,k}^-$ is computed as

$$\boldsymbol{S}_{zz,k}^- = \text{hhr}\left\{ \left[\begin{array}{c|c} (\boldsymbol{z}_k^{-(1:r)} \ominus \boldsymbol{m}_{z,k}^-)\boldsymbol{W}^{(1:r)} & (\boldsymbol{z}_k^{-(r+1:q_k^-)} \ominus \boldsymbol{m}_{z,k}^-)\boldsymbol{W}^{(r+1:q_k^-)} \end{array} \right] \right\} ,$$

where, as with $\bar{\boldsymbol{x}}_k^{-(i:j)}$ in the previous section, $\boldsymbol{z}_k^{-(i:j)}$ denotes a column-wise concatenation of the the i^{th} to j^{th} measurement-transformed quadrature points and that all quadrature points have been sorted such that points $\boldsymbol{z}_k^{-(r+1:q_k^-)}$ are the points with negative weights. Similar to what was presented in the propagation stage, this can be accomplished using the provided Algorithm 4 via

$$\boldsymbol{S}_{zz,k}^- = \text{HHR\_UP\_AND\_DOWNDATE}\left\{\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{0}_{n_z\times 1}\right\} ,$$

where

$$\boldsymbol{U}_1 = (\boldsymbol{z}_k^{-(r+1:q_k^-)} \ominus \boldsymbol{m}_{z,k}^-)\boldsymbol{W}^{(r+1:q_k^-)}$$

$$\boldsymbol{U}_2 = \begin{cases} (\boldsymbol{z}_k^{-(1:r)} \ominus \boldsymbol{m}_{z,k}^-)\boldsymbol{W}^{(1:r)} & \text{if } r > 0 \\ \boldsymbol{0}_{n_z \times 1} & \text{otherwise} \end{cases}.$$

The cross covariances, $\boldsymbol{P}_{xz,k}^-$ and $\boldsymbol{P}_{cz,k}^-$, are then found from Eqs. (3.17c) and (3.17d), respectively, and the gains, $\boldsymbol{K}_{x,k}$ and $\boldsymbol{K}_{c,k}$, are found using Eqs. (3.23).

Then, the quadrature-based square-root factor update is completed using Eq. (3.27). Notably, this is precisely the same square-root factor update as with the linearization-based method of Section 3.2.2.2. Once the expectations in Eqs. (3.11) are computed, the same linear update equations apply for both the linearization- and quadrature-based methods. This leads to an important theoretical point: the linearization- and quadrature-based filters are not intrinsically different estimation mechanisms, as they are *both* based on the same LMMSE estimation principles and utilize the very same formulation in terms of the expectation operations. They differ, however, in the approximations use to evaluate those expectations, and therein lie the differences that produce differences in filter performance.

Summary. The quadrature-based, square-root formulation of the MMSE consider filter is summarized in Table 3.5.

3.2.4. Improvements to Standard Square-Root Filters. These new developments also permit improvements to the design of the standard (i.e. without consider parameters) square-root filters described in Section 2.3. Specifically, (i) improvements can be made to the square-root filter's corrector step to enable a full rank downdate, rather than a sequence of rank-1 Cholesky downdates, and (ii) the quadrature-based square-root filter's two-step procedure to account for negative weights can be reformulated as a single step. Nomenclature is taken directly from that Section 2.3, so a reader is asked to refer back to those pages if clarity is required.

Improvements to Standard Square-Root Correctors. In the corrector step for the standard square-root filters of Section 2.3, it was found that, when processing a measurement $\boldsymbol{z}_k$, it is necessary to perform n_z rank-1 Cholesky downdates. That is, the

Table 3.5. Quadrature-based, square-root formulation of the MMSE consider filter.

Models

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{w}_{k-1}) \qquad \boldsymbol{P}_{ww,k-1} = \boldsymbol{S}_{ww,k-1}\boldsymbol{S}_{ww,k-1}^T = \mathbb{E}\left\{\boldsymbol{w}_{k-1}\boldsymbol{w}_{k-1}^T\right\}$$
$$\boldsymbol{c}_k = \boldsymbol{g}(\boldsymbol{c}_{k-1}, \boldsymbol{u}_{k-1}) \qquad \boldsymbol{P}_{uu,k-1} = \boldsymbol{S}_{uu,k-1}\boldsymbol{S}_{uu,k-1}^T = \mathbb{E}\left\{\boldsymbol{u}_{k-1}\boldsymbol{u}_{k-1}^T\right\}$$
$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{c}_k, \boldsymbol{v}_k) \qquad \boldsymbol{P}_{vv,k} = \boldsymbol{S}_{vv,k}\boldsymbol{S}_{vv,k}^T = \mathbb{E}\left\{\boldsymbol{v}_k\boldsymbol{v}_k^T\right\}$$

Initialization

$$\boldsymbol{m}_{x,0} = \mathbb{E}\left\{\boldsymbol{x}_0\right\} \qquad \boldsymbol{P}_{xx,0} = \boldsymbol{S}_{xx,0}\boldsymbol{S}_{xx,0}^T + \boldsymbol{S}_{xc,0}\boldsymbol{S}_{xc,0}^T$$
$$\boldsymbol{m}_{c,0} = \mathbb{E}\left\{\boldsymbol{c}_0\right\} \qquad \boldsymbol{P}_{cc,0} = \boldsymbol{S}_{cc,0}\boldsymbol{S}_{cc,0}^T$$

Predictor

Draw points: $\boldsymbol{x}_{k-1}^{+(\ell)}$, $\boldsymbol{c}_{k-1}^{+(\ell)}$, $\boldsymbol{w}_{k-1}^{(\ell)}$, and $\boldsymbol{u}_{k-1}^{(\ell)}$ with weights $w_{m,k-1}^{+(\ell)}, w_{c,k-1}^{+(\ell)}$ for $\ell \in \{1, \ldots, q_{k-1}^+\}$.

$$\boldsymbol{x}_k^{-(\ell)} = \boldsymbol{f}(\boldsymbol{x}_{k-1}^{+(\ell)}, \boldsymbol{c}_{k-1}^{+(\ell)}, \boldsymbol{w}_{k-1}^{(\ell)})$$
$$\boldsymbol{c}_k^{-(\ell)} = \boldsymbol{g}(\boldsymbol{c}_{k-1}^{+(\ell)}, \boldsymbol{u}_{k-1}^{(\ell)}) \qquad \bar{\boldsymbol{S}}_k^- = \text{HHR\_UP\_AND\_DOWNDATE}\left\{\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{0}_{\bar{n}\times 1}\right\}$$
$$\boldsymbol{m}_{x,k}^- = \textstyle\sum_{\ell=1}^{q_{k-1}^+} w_{m,k-1}^{+(\ell)} \boldsymbol{x}_k^{-(\ell)} \qquad \boldsymbol{U}_1 = (\bar{\boldsymbol{x}}_k^{-(r+1:q_{k-1}^+)} \ominus \bar{\boldsymbol{m}}_k^-)\boldsymbol{W}^{(r+1:q_{k-1}^+)}$$
$$\boldsymbol{m}_{c,k}^- = \textstyle\sum_{\ell=1}^{q_{k-1}^+} w_{m,k-1}^{+(\ell)} \boldsymbol{c}_k^{-(\ell)} \qquad \boldsymbol{U}_2 = \begin{cases} (\bar{\boldsymbol{x}}_k^{-(1:r)} \ominus \bar{\boldsymbol{m}}_k^-)\boldsymbol{W}^{(1:r)} & \text{if } r > 0 \\ \boldsymbol{0}_{\bar{n}\times 1} & \text{otherwise} \end{cases}$$

Corrector

Draw points: $\boldsymbol{x}_k^{-(\ell)}$, $\boldsymbol{c}_k^{-(\ell)}$, and $\boldsymbol{v}_k^{(\ell)}$ with weights $w_{m,k}^{-(\ell)}, w_{c,k}^{-(\ell)}$ for $\ell \in \{1, \ldots, q_k^-\}$.

$$\boldsymbol{m}_{x,k}^+ = \boldsymbol{m}_{x,k}^- + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^-)$$
$$\boldsymbol{m}_{c,k}^+ = \boldsymbol{m}_{c,k}^- \qquad \boldsymbol{K}_{x,k} = \left[\boldsymbol{P}_{xz,k}^-(\boldsymbol{S}_{zz,k}^-)^{-T}\right](\boldsymbol{S}_{zz,k}^-)^{-1} \qquad \bar{\boldsymbol{K}}_k = \begin{bmatrix}\boldsymbol{K}_{x,k} \\ \boldsymbol{K}_{c,k}\end{bmatrix}$$
$$\boldsymbol{z}_k^{-(\ell)} = \boldsymbol{h}(\boldsymbol{x}_k^{-(\ell)}, \boldsymbol{c}_k^{-(\ell)}, \boldsymbol{v}_k^{(\ell)}) \qquad \boldsymbol{K}_{c,k} = \left[\boldsymbol{P}_{cz,k}^-(\boldsymbol{S}_{zz,k}^-)^{-T}\right](\boldsymbol{S}_{zz,k}^-)^{-1} \qquad \boldsymbol{\Upsilon} = \begin{bmatrix}\boldsymbol{0}_{n_x\times n_c} \\ \boldsymbol{I}_{n_c\times n_c}\end{bmatrix}$$
$$\boldsymbol{m}_{z,k}^- = \textstyle\sum_{\ell=1}^{q_k^-} w_{m,k}^{-(\ell)} \boldsymbol{z}_k^{-(\ell)}$$
$$\boldsymbol{P}_{xz,k}^- = \textstyle\sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)} (\boldsymbol{x}_k^{-(\ell)} - \boldsymbol{m}_{x,k}^-)(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)^T$$
$$\boldsymbol{P}_{cz,k}^- = \textstyle\sum_{\ell=1}^{q_k^-} w_{c,k}^{-(\ell)} (\boldsymbol{c}_k^{-(\ell)} - \boldsymbol{m}_{c,k}^-)(\boldsymbol{z}_k^{-(\ell)} - \boldsymbol{m}_{z,k}^-)^T$$
$$\bar{\boldsymbol{S}}_k^+ = \text{HHR\_UP\_AND\_DOWNDATE}\left\{\bar{\boldsymbol{S}}_k^-,\ \bar{\boldsymbol{K}}_k\boldsymbol{S}_{zz,k}^-,\ \boldsymbol{\Upsilon}\boldsymbol{K}_{c,k}\boldsymbol{S}_{zz,k}^-\right\}$$
$$\boldsymbol{S}_{zz,k}^- = \text{HHR\_UP\_AND\_DOWNDATE}\left\{\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{0}_{n_z\times 1}\right\}$$
$$\boldsymbol{U}_1 = (\boldsymbol{z}_k^{-(r+1:q_k^-)} \ominus \boldsymbol{m}_{z,k}^-)\boldsymbol{W}^{(r+1:q_k^-)}$$
$$\boldsymbol{U}_2 = \begin{cases} (\boldsymbol{z}_k^{-(1:r)} \ominus \boldsymbol{m}_{z,k}^-)\boldsymbol{W}^{(1:r)} & \text{if } r > 0 \\ \boldsymbol{0}_{n_z\times 1} & \text{otherwise} \end{cases}$$

sequence of rank-1 modifications

$$\boldsymbol{S}_{xx,k}^+(\boldsymbol{S}_{xx,k}^+)^T = \boldsymbol{S}_{xx,k}^-(\boldsymbol{S}_{xx,k}^-)^T - \boldsymbol{u}_1\boldsymbol{u}_1^T - \cdots - \boldsymbol{u}_{n_z}\boldsymbol{u}_{n_z}^T$$

was required. Instead, return to the full, rank-n_z modification

$$\boldsymbol{S}_{xx,k}^+(\boldsymbol{S}_{xx,k}^+)^T = \boldsymbol{S}_{xx,k}^-(\boldsymbol{S}_{xx,k}^-)^T - \boldsymbol{U}_k\boldsymbol{U}_k^T\,.$$

The updated square-root factor $\boldsymbol{S}_{xx,k}^+$ can be obtained using a sequence of hyperbolic Householder reflections on

$$\boldsymbol{S}_{xx,k}^+ = \text{hhr}\left\{\left[\begin{array}{c|c} \boldsymbol{U}_k & \boldsymbol{S}_{xx,k}^- \end{array}\right]\right\}\,,$$

where the reflections are taken with respect to the signature matrix

$$\boldsymbol{Y} = \text{blkdiag}\{-\boldsymbol{I}_{n_z \times n_z}, \boldsymbol{I}_{n_x \times n_x}\}.$$

This operation can be performed using Algorithm 4 as

$$\boldsymbol{S}^{+}_{xx,k} = \text{HHR\_UP\_AND\_DOWNDATE}\left\{\boldsymbol{S}^{-}_{xx,k}, \boldsymbol{U}_k, \boldsymbol{0}_{n_x,1}\right\}.$$

Note that rather than performing a sequence of rank-1 downdates, the full, rank-n_z downdate is performed in a single pass.

Improvements to Standard, Quadrature-based Square-Root Filters. In Section 2.3.3, it was observed that, to account for potentially negative quadrature weights, a quadrature-based square-root filter requires a two-step procedure to compute both $\boldsymbol{S}^{-}_{xx,k}$ and $\boldsymbol{S}^{-}_{zz,k}$: an RQ-decomposition for the positive weights and a Cholesky downdate for the negative weights. A more elegant procedure is desired, and hyperbolic Householder reflections offer a candidate option.

Recalling that the predicted covariance can be written in terms of square-root factors of the form

$$\boldsymbol{S}^{-}_{xx,k}(\boldsymbol{S}^{-}_{xx,k})^T = \sum_{\ell=1}^{q^{+}_{k-1}} w^{+(\ell)}_{c,k-1}(\boldsymbol{x}^{-(\ell)}_k - \boldsymbol{m}^{-}_{x,k})(\boldsymbol{x}^{-(\ell)}_k - \boldsymbol{m}^{-}_{x,k})^T,$$

it's easy to see that since the first r weights are negative, the sum can be partitioned as

$$\begin{aligned}\boldsymbol{S}^{-}_{xx,k}(\boldsymbol{S}^{-}_{xx,k})^T = &\sum_{\ell=r+1}^{q^{+}_{k-1}} w^{+(\ell)}_{c,k-1}(\boldsymbol{x}^{-(\ell)}_k - \boldsymbol{m}^{-}_{x,k})(\boldsymbol{x}^{-(\ell)}_k - \boldsymbol{m}^{-}_{x,k})^T \\ &- \sum_{\ell=1}^{r} |w^{+(\ell)}_{c,k-1}|(\boldsymbol{x}^{-(\ell)}_k - \boldsymbol{m}^{-}_{x,k})(\boldsymbol{x}^{-(\ell)}_k - \boldsymbol{m}^{-}_{x,k})^T.\end{aligned}$$

Following the same procedure as in the previous section, it is relatively straightforward to demonstrate that hyperbolic Householder reflections can be used to compute $\boldsymbol{S}^{-}_{xx,k}$ as

$$\boldsymbol{S}^{-}_{xx,k} = \text{HHR\_UP\_AND\_DOWNDATE}\{\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{0}_{\bar{n},1}\}$$

where

$$\boldsymbol{U}_1 = (\boldsymbol{x}_k^{-(r+1:q_{k-1}^+)} - \boldsymbol{m}_{x,k}^-)\boldsymbol{W}^{(r+1:q_{k-1}^+)}$$
$$\boldsymbol{U}_2 = \begin{cases} (\boldsymbol{x}_k^{-(1:r)} - \boldsymbol{m}_{x,k}^-)\boldsymbol{W}^{(1:r)} & \text{if } r > 0 \\ \boldsymbol{0}_{\bar{n}\times 1} & \text{otherwise} \end{cases}.$$

The same approach can be applied to computing $\boldsymbol{S}_{zz,k}^-$, and this follows in the same way.

3.2.5. Efficient Implementations with ECRVs. This section aims to exploit unique characteristics of the linearization-based (i.e. extended) consider square-root propagation procedure outlined in Section 3.2.2 to produce a more computationally efficient time update. The motivation for looking specifically at the linearization-based procedure is twofold. First, linearization is by far the most widely adopted technique for accommodation of nonlinearities in real-time filtering applications (e.g. navigation) where computational efficiency is of principle interest. Second, the linearization-based formulation offers convenient manipulation and exposition via concatenation. Similar improvements in efficiency can be employed for the quadrature-based methods presented in this section but are omitted since a reader familiar with the presented approach should be able to handily produce a similar procedure for those methods.

Using the matrix definitions from Section 3.2.2.1 and defining

$$\bar{\boldsymbol{P}}_{ww,k-1} = \begin{bmatrix} \boldsymbol{P}_{ww,k-1} & \boldsymbol{0}_{n_x\times n_c} \\ \boldsymbol{0}_{n_c\times n_x} & \boldsymbol{0}_{n_c\times n_c} \end{bmatrix}$$
$$\bar{\boldsymbol{P}}_{uu,k-1} = \begin{bmatrix} \boldsymbol{0}_{n_x\times n_x} & \boldsymbol{0}_{n_x\times n_c} \\ \boldsymbol{0}_{n_c\times n_x} & \boldsymbol{P}_{uu,k-1} \end{bmatrix}$$

permits Eq. (3.21) to be rewritten as

$$\bar{\boldsymbol{P}}_k^- = \underbrace{\bar{\boldsymbol{G}}_{k-1}\underbrace{\left[\bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{P}}_{k-1}^+\bar{\boldsymbol{F}}_{k-1}^T + \bar{\boldsymbol{G}}_{k-1}^{-1}\bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{P}}_{ww,k-1}\bar{\boldsymbol{M}}_{k-1}^T\bar{\boldsymbol{G}}_{k-1}^{-T}\right]}_{1^{\text{st}}\text{ stage, produces }\bar{\boldsymbol{P}}_k^*=\bar{\boldsymbol{S}}_k^*(\bar{\boldsymbol{S}}_k^*)^T}\bar{\boldsymbol{G}}_{k-1}^T + \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{P}}_{uu,k-1}\bar{\boldsymbol{M}}_{k-1}^T}_{2^{\text{nd}}\text{ stage}}$$

In this case, Eq. (3.21) has been rewritten to contain two "stages," with the first stage pertaining principally to the motion of the state and the second stage utilizing the result of the first stage to account for motion of the parameters. The idea is to perform a lower-dimensional square-root time update according to the typically complex dynamical system governing motion of the state and a second, much simpler, square-root time update for the consider parameters. This will become useful when certain modeling decisions are made for the parameters, $\boldsymbol{c}_k$, particularly that the corresponding dynamical noise covariance matrix $\boldsymbol{P}^-_{uu,k}$ is diagonal for all k. This result will hold for any such parameters, but a particularly convenient and efficient formulation is produced when the parameters are modeled as exponentially correlated random variables (ECRVs) [50]. The following discussion details a more efficient algorithm under these modeling choices.

The First Stage: State Time Update. The first stage of the time update produces the square-root factor of $\bar{\boldsymbol{P}}^*_k$, where

$$\bar{\boldsymbol{P}}^*_k = \bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{P}}^+_{k-1}\bar{\boldsymbol{F}}^T_{k-1} + \bar{\boldsymbol{G}}^{-1}_{k-1}\bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{P}}_{ww,k-1}\bar{\boldsymbol{M}}^T_{k-1}\bar{\boldsymbol{G}}^{-T}_{k-1}\,.$$

As before, substituting in square-root factors of covariance and factoring produces

$$\bar{\boldsymbol{S}}^*_k(\bar{\boldsymbol{S}}^*_k)^T = \left[\begin{array}{c|c} \bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{S}}^+_{k-1} & \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{S}}_{ww,k-1} \end{array}\right]\left[\begin{array}{c|c} \bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{S}}^+_{k-1} & \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{S}}_{ww,k-1} \end{array}\right]^T, \tag{3.31}$$

since the required product reduces to

$$\bar{\boldsymbol{G}}^{-1}_{k-1}\bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{S}}_{ww,k-1} = \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{S}}_{ww,k-1}\,,$$

where

$$\bar{\boldsymbol{S}}_{ww,k-1} = \begin{bmatrix} \boldsymbol{S}_{ww,k-1} & \boldsymbol{0}_{n_x\times n_c} \\ \boldsymbol{0}_{n_c\times n_x} & \boldsymbol{0}_{n_c\times n_c} \end{bmatrix}.$$

In practice, for efficiency and stability, matrix inverses are best avoided. The inverse $\bar{\boldsymbol{G}}^{-1}_{k-1}$ has been avoided entirely, requiring only that the dynamics of $\boldsymbol{c}_k$ do not depend on $\boldsymbol{x}_k$ (an assumption already made in all previous discussion). Recalling that the partitioned,

concatenated square-root factor takes the form

$$\bar{\boldsymbol{S}}_{k-1}^{+} = \begin{bmatrix} \boldsymbol{S}_{xx,k-1}^{+} & \boldsymbol{S}_{xc,k-1}^{+} \\ \boldsymbol{0}_{n_c \times n_x} & \boldsymbol{S}_{cc,k-1}^{+} \end{bmatrix}$$

and multiplying out Eq. (3.31), it is observed that the first stage can be written to produce[4]

$$\boldsymbol{S}_{xx,k}^{*} = \mathrm{rq}\left\{ \left[\begin{array}{c|c} \boldsymbol{F}_{x,k-1}\boldsymbol{S}_{xx,k-1}^{+} & \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1} \end{array} \right] \right\} \tag{3.32a}$$

$$\boldsymbol{S}_{xc,k}^{*} = \boldsymbol{F}_{x,k-1}\boldsymbol{S}_{xc,k-1}^{+} + \boldsymbol{F}_{c,k-1}\boldsymbol{S}_{cc,k-1}^{+} \tag{3.32b}$$

$$\boldsymbol{S}_{cc,k}^{*} = \boldsymbol{S}_{cc,k-1}^{+} \,. \tag{3.32c}$$

As desired, this first stage of the propagation only considers the dynamical motion of the state and leaves the dynamical motion of the parameters to the second stage. Proof is given in Appendix C.4.

Observe that, instead of performing a RQ-decomposition on the augmented term

$$\left[\begin{array}{c|c} \bar{\boldsymbol{F}}_{k-1}\bar{\boldsymbol{S}}_{k-1}^{+} & \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{S}}_{ww,k-1} \end{array} \right]$$

in Eq. (3.31) to produce $\bar{\boldsymbol{S}}_{k}^{*}$, a lower-dimensional RQ-decomposition is used in Eq. (3.32a), and the remaining required terms are solved for using Eqs. (3.32b) and (3.32c). This lower-dimensioned RQ-decomposition offers advantages in terms of required computational effort and, if the second stage can be performed efficiently, translates into substantial savings in required computation time for the consider square-root filter's propagation step.

Remark 3.1 (The Case for Upper Triangular Factors)**.** *It is at this point that the value in selecting upper triangular factors, versus lower triangular factors, becomes apparent despite an inclination to presume they are equivalent. Indeed, on the surface, the choice of upper versus lower triangular factor appears superficial, but to illustrate how this is not the case,*

[4]The RQ-decomposition required in Eq. (3.32a) can be accomplished, as before, with the $\mathrm{HR}\{\cdot\}$ algorithm in Algorithm 2.

suppose that the factors were instead lower triangular of the form

$$\bar{\boldsymbol{L}}_{k-1}^{+} = \begin{bmatrix} \boldsymbol{S}_{xx,k-1}^{+} & \boldsymbol{0}_{n_x \times n_c} \\ \boldsymbol{S}_{xc,k-1}^{+} & \boldsymbol{S}_{cc,k-1}^{+} \end{bmatrix}.$$

Then, multiplying out Eq. (3.31) yields

$$\bar{\boldsymbol{S}}_{k}^{*}(\bar{\boldsymbol{S}}_{k}^{*})^{T} = \begin{bmatrix} \boldsymbol{F}_{x,k-1}\boldsymbol{S}_{xx,k-1}^{+} + \boldsymbol{F}_{c,k-1}\boldsymbol{S}_{xc,k-1}^{+} & \boldsymbol{F}_{c,k-1}\boldsymbol{S}_{cc,k-1}^{+} & \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1} & \boldsymbol{0}_{n_x \times n_c} \\ \boldsymbol{S}_{xc,k-1}^{+} & \boldsymbol{S}_{cc,k-1}^{+} & \boldsymbol{0}_{n_c \times n_x} & \boldsymbol{0}_{n_c \times n_c} \end{bmatrix} \begin{bmatrix} \ldots \end{bmatrix}^{T}.$$

Notice that, in contrast to the upper triangular case, the bottom-left element of the result is not a matrix of zeros; instead, it is the term $\boldsymbol{S}_{xc,k-1}^{+}$*. Further multiplying out this expression to form some* $\boldsymbol{A}_1$*,* $\boldsymbol{A}_2$*, and* $\boldsymbol{A}_3$*, as done in the proof in Appendix C.4, produces a series of cross-terms, such as the product* $\boldsymbol{S}_{xx,k-1}^{+}(\boldsymbol{S}_{xc,k-1}^{+})^{T}$ *that prevents the desired separability between the state and consider parameters. This observation serves as an important reminder that some decisions, no matter how superficial or inconsequential they may appear, can have tremendous implications in surprising places.*

The Second Stage: Parameter Time Update. With the first stage complete, the remaining problem becomes, in terms of concatenated square-root factors,

$$\bar{\boldsymbol{S}}_{k}^{-}(\bar{\boldsymbol{S}}_{k}^{-})^{T} = \bar{\boldsymbol{G}}_{k-1}\bar{\boldsymbol{S}}_{k}^{*}(\bar{\boldsymbol{S}}_{k}^{*})^{T}\bar{\boldsymbol{G}}_{k-1}^{T} + \bar{\boldsymbol{M}}_{k-1}\bar{\boldsymbol{S}}_{uu,k-1}\bar{\boldsymbol{S}}_{uu,k-1}^{T}\bar{\boldsymbol{M}}_{k-1}^{T}, \tag{3.33}$$

where

$$\bar{\boldsymbol{S}}_{uu,k-1} = \begin{bmatrix} \boldsymbol{0}_{n_x \times n_x} & \boldsymbol{0}_{n_x \times n_c} \\ \boldsymbol{0}_{n_c \times n_x} & \boldsymbol{S}_{uu,k-1} \end{bmatrix}.$$

Ultimately, to complete the time update, the second stage must produce the terms $\boldsymbol{S}_{xx,k}^{-}$, $\boldsymbol{S}_{xc,k}^{-}$, and $\boldsymbol{S}_{cc,k}^{-}$, all elements of the concatenated propagated square-root factor $\bar{\boldsymbol{S}}_{k}^{-}$. Equation (3.33) implies that

$$\begin{aligned} \boldsymbol{S}_{xx,k}^{-}(\boldsymbol{S}_{xx,k}^{-})^{T} + \boldsymbol{S}_{xc,k}^{-}(\boldsymbol{S}_{xc,k}^{-})^{T} &= \boldsymbol{S}_{xx,k}^{*}(\boldsymbol{S}_{xx,k}^{*})^{T} + \boldsymbol{S}_{xc,k}^{*}(\boldsymbol{S}_{xc,k}^{*})^{T} \\ \boldsymbol{S}_{xc,k}^{-}(\boldsymbol{S}_{cc,k}^{-})^{T} &= \boldsymbol{S}_{xc,k}^{*}(\boldsymbol{G}_{u,k-1}\boldsymbol{S}_{cc,k}^{*})^{T} \\ \boldsymbol{S}_{cc,k}^{-}(\boldsymbol{S}_{cc,k}^{-})^{T} &= \boldsymbol{G}_{c,k-1}\boldsymbol{S}_{cc,k}^{*}(\boldsymbol{S}_{cc,k}^{*})^{T}\boldsymbol{G}_{c,k-1}^{T} + \boldsymbol{G}_{u,k-1}\boldsymbol{S}_{uu,k-1}\boldsymbol{S}_{uu,k-1}^{T}\boldsymbol{G}_{u,k-1}^{T}, \end{aligned}$$

and therefore,

$$\bar{\boldsymbol{S}}_k^- = \mathrm{rq}\left\{\left[\begin{array}{c|c} \bar{\boldsymbol{G}}_{k-1}\bar{\boldsymbol{S}}_{k-1}^* & \boldsymbol{U} \end{array}\right]\right\}, \tag{3.34}$$

where

$$\boldsymbol{U} = \mathrm{blkdiag}\{\boldsymbol{0}_{n_x \times n_x},\ \boldsymbol{G}_{u,k-1}\boldsymbol{S}_{uu,k-1}\}\,.$$

In the interest of obtaining a more efficient implementation, a method to avoid the RQ-decomposition in Eq. (3.34) is presented. To that end, assume that the parameters are uncorrelated with each other through time (i.e. $\boldsymbol{P}_{uu,k-1}$ is diagonal for all k, and therefore so is $\boldsymbol{S}_{uu,k-1}$). Accordingly, $\boldsymbol{G}_{u,k-1}$ must also be diagonal, and, therefore, $\boldsymbol{G}_{u,k-1}\boldsymbol{S}_{uu,k-1}$ is diagonal. Instead of performing an RQ-decomposition, a more efficient implementation that exploits the diagonal nature of $\boldsymbol{G}_{u,k-1}\boldsymbol{S}_{uu,k-1}$ can be achieved using Algorithm 8 in Appendix A, where

$$\bar{\boldsymbol{S}}_k^- = \textsc{chol\_diag\_update}\left\{\bar{\boldsymbol{G}}_{k-1}\bar{\boldsymbol{S}}_{k-1}^*,\ \boldsymbol{U}\right\}. \tag{3.35}$$

This update is here called the "Cholesky diagonal update" because it resembles a Cholesky update, but, instead of performing n_c rank-1 Cholesky updates, the full rank-n_c update is performed in one pass thanks to the diagonal nature of the update term(s). By directly exploiting the diagonal nature of the update term, Algorithm 8 very efficiently computes the time-updated square-root factor corresponding to the consider parameters. This is a nonstandard matrix modification, but the author is hesitant to make any claims regarding the novelty of this update. Nonetheless, the author is unaware of its existence or representation within the literature, and so it has been named the Cholesky diagonal update and is presented in Appendix A.

Summary of Efficient Time Update. Given some $\bar{\boldsymbol{S}}_{k-1}^+$, in the case of consider parameters that are timewise uncorrelated with one another (i.e. $\boldsymbol{G}_{u,k-1}$ and $\boldsymbol{S}_{uu,k-1}$ are diagonal), the first stage of the propagation is solved with Eqs. (3.32) and the result is used to solve Eq. (3.35), completing the efficient time update.

Timing study. To evaluate the reduction in runtime afforded by the efficient implementation, the "full" time update using Eq. (3.22) is compared to the two-stage efficient time update given by Eqs. (3.32) and Eq. (3.35). For this study, the state is fixed to be of dimension $n_x = 4$, and the consider parameter dimension, n_c, is varied from 5 to 100. Representative system matrices are selected as

$$\boldsymbol{F}_{x,k-1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{F}_{c,k-1} = \begin{bmatrix} \sqrt{\Delta t} \cdot \boldsymbol{I}_{4\times 4} & \boldsymbol{0}_{4\times(n_c-4)} \end{bmatrix},$$

$\boldsymbol{F}_{w,k-1} = \boldsymbol{I}_{4\times 4}$, $\boldsymbol{G}_{c,k-1} = \exp\{-\Delta t\} \cdot \boldsymbol{I}_{n_c\times n_c}$, $\boldsymbol{G}_{u,k-1} = \boldsymbol{I}_{n_c\times n_c}$, $\boldsymbol{S}_{ww,k-1} = \boldsymbol{I}_{4\times 4}$, $\boldsymbol{S}_{uu,k-1} = \boldsymbol{I}_{n_c\times n_c}$, and $\Delta t = 0.1$. Note that this design conforms with all of the necessary modeling assumptions of the efficient implementation (that is, independent consider parameters that are timewise uncorrelated with one another).

For a given n_c, a random $(4 + n_c) \times (4 + n_c)$ covariance matrix is constructed by filling some $(4 + n_c) \times (4 + n_c)$ matrix $\boldsymbol{P}$ with random samples from a standard uniform on $(0, 1)$. The matrix is forced to symmetry by computing $\boldsymbol{P} \leftarrow (\boldsymbol{P} + \boldsymbol{P}^T)/2$, and, since every $a_{ij} \in \boldsymbol{P}$ obeys $a_{ij} < 1$ and any symmetric, diagonally dominant matrix is positive definite, positivity of $\boldsymbol{P}$ is guaranteed by computing $\boldsymbol{P} \leftarrow \boldsymbol{P} + (4 + n_c)\boldsymbol{I}_{(4+n_c)\times(4+n_c)}$. Then, $\boldsymbol{P}$ is guaranteed to be a SPD to serve as an analog to fully populated covariance matrix for the timing study. However, the efficient implementation requires that the consider parameters be independent, and $\boldsymbol{P}$ is "full" of the form

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{P}_{xx} & \boldsymbol{P}_{xc} \\ \boldsymbol{P}_{xc}^T & \boldsymbol{P}_{cc} \end{bmatrix}.$$

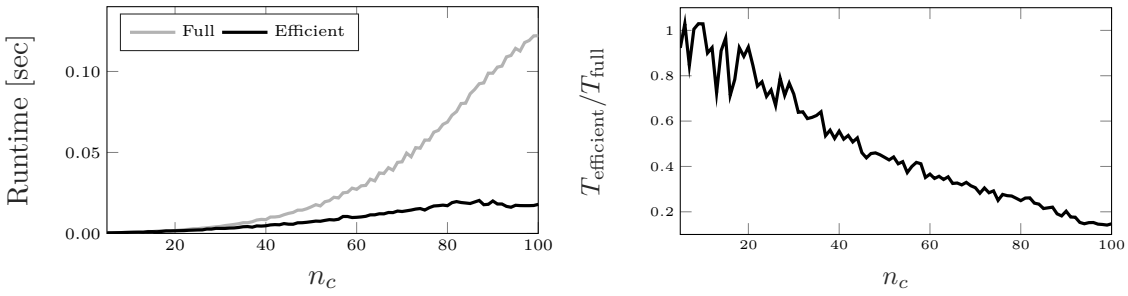
The parameter term $\boldsymbol{P}_{cc}$ is replaced by $\boldsymbol{P}_{cc} \leftarrow \text{diag-only}\{\boldsymbol{P}_{cc}\}$, where diag-only$\{\cdot\}$ is used to denote an operator that sets all entries but the diagonals to zero. Therefore, $\boldsymbol{P}$ is then a matrix that is full except that $\boldsymbol{P}_{cc}$ is diagonal, satisfying the requirements of the efficient implementation. Then, the upper triangular Cholesky factor is computed such

that $\boldsymbol{P} = \boldsymbol{S}\boldsymbol{S}^T$, where

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{S}_{xx} & \boldsymbol{S}_{xc} \\ \boldsymbol{0}_{n_c \times 4} & \boldsymbol{S}_{cc} \end{bmatrix}$$

is used as the representative square-root factor analog in these studies.

As n_c is varied, the full and efficient implementations are both computed 10,000 times and the runtimes to execute each are recorded. Then, the average runtime for dimension n_c is computed by dividing the total runtime of each implementation by 10,000. The runtime for both methods as n_c increases is presented in Figure 3.1a, and the ratio of the average time of the efficient procedure to the full procedure (interpreted as a cost reduction factor) is presented in Figure 3.1b. It can be seen that the efficient procedure can be expected to offer speed improvements across the board, at worst requiring the same as the full procedure. However, as n_c increases, i.e. the number of considered parameters increases, particularly as it exceeds 20, the performance advantages of the efficient procedure become enormous. At the largest dimension of $n_c = 100$, the efficient procedure requires a mere 15% the computing time that the full procedure requires. Therefore, the efficient formulation is an excellent improvement in estimator design for problems that possess a large number of parameters, such as many practical navigation problems, estimation in complex and uncertain gravity fields with many uncertain coefficients, analysis of building structural loading, weather forecasting, or network traffic predictions, to name a few.



(a) (b)

Figure 3.1. Timing results as consider parameter vector dimension (n_c) changes between the full and efficient time update formulations.

Remark 3.2 (Parameters Without Dynamic Noise)**.** *In the case of consider parameters that have no noise in their time evolution (i.e.* $\boldsymbol{u}_k = \boldsymbol{0}\ \forall\ k \geq 0$*, or equivalently* $\boldsymbol{P}_{uu,k-1} = \boldsymbol{0}_{n_u \times n_u}$*), the RQ triangularization/Cholesky diagonal update can be skipped entirely, and instead* $\boldsymbol{S}^{-}_{xx,k} = \boldsymbol{S}^{*}_{xx,k}$*,* $\boldsymbol{S}^{-}_{xc,k} = \boldsymbol{S}^{*}_{xc,k}$*, and* $\boldsymbol{S}^{-}_{cc,k} = \boldsymbol{G}_{c,k-1}\boldsymbol{S}^{*}_{cc,k}$*.*

Remark 3.3 (Parameter Modeling using ECRVs)**.** *One common parameter modeling choice that has gained a great deal of popularity due to its practical advantages is that of treating the parameters as ECRVs, or random variables resulting from a first-order Gauss-Markov process [50]. Indeed, this modeling choice has seen widespread use within spaceflight, including, but certainly not limited to, navigators' use for Apollo [19], the Space Shuttle [31], and the Orion vehicle [30, 49]. These parameters have the tremendous benefit of having deterministic dynamics as well as being stochastically bounded by a user-specified steady-state variance. ECRVs have a broad applicability to a large number of parameter modeling problems, such as range sensing biases, accelerometer/gyroscope scale factor and misalignment biases, or sensors with lengthy transient sensing errors (such as on startup or reset).*

These ECRVs are timewise correlated with themselves but are uncorrelated through time with other ECRVs. Therefore, all the requirements of diagonality presented in this section to produce an efficient implementation are satisfied. In this case, the required matrices become

$$\begin{aligned}\boldsymbol{G}_{c,k-1} &= \operatorname{diag}\left\{\exp\left\{-\frac{\Delta t_k}{\tau_1}\right\}, \ldots, \exp\left\{-\frac{\Delta t_k}{\tau_{n_c}}\right\}\right\} \\ \boldsymbol{G}_{u,k-1} &= \boldsymbol{I}_{n_c \times n_c}\,.\end{aligned}$$

where $\Delta t_k = t_k - t_{k-1}$ *and* τ_i *is the time constant of the* i^{th} *consider parameter. Additionally,*

$$\boldsymbol{P}_{uu,k} = \operatorname{diag}\left\{\left(1 - \exp\left\{\frac{-2\Delta t_k}{\tau_1}\right\}\right)\sigma_1^2, \ldots, \left(1 - \exp\left\{\frac{-2\Delta t_k}{\tau_{n_c}}\right\}\right)\sigma_{n_c}^2\right\}\,,$$

where σ_i^2 *is the steady-state value of the variance of the* i^{th} *consider parameter, and therefore*

$$\boldsymbol{S}_{uu,k} = \operatorname{diag}\left\{\sigma_1\sqrt{1 - \exp\left\{\frac{-2\Delta t_k}{\tau_1}\right\}}, \ldots, \sigma_{n_c}\sqrt{1 - \exp\left\{\frac{-2\Delta t_k}{\tau_{n_c}}\right\}}\right\}\,.$$

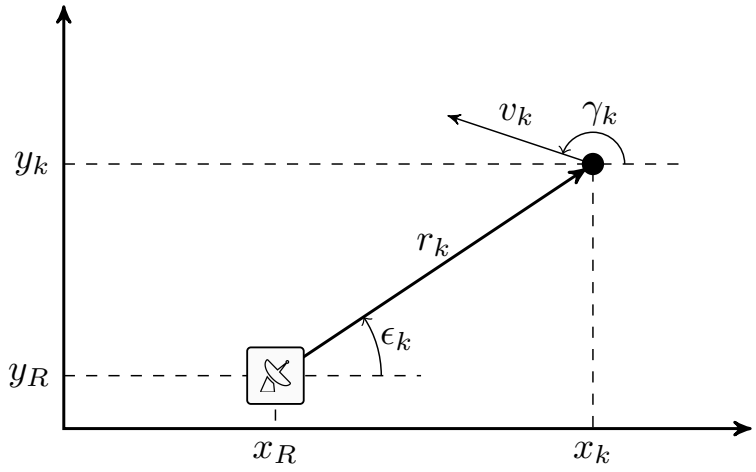


Figure 3.2. Schematic of a radar tracking the ballistic trajectory.

3.2.6. Numerical Example. To assess the performance of the described consider filtering procedures and to illustrate their advantages over standard implementations, a numerical study based on works by Farina [65] and Ristic [66] is presented. This scenario is used several times throughout this dissertation and is repeatedly referred to as "the ballistic trajectory" example.

Consider an object on a ballistic trajectory that is re-entering the atmosphere, and during this re-entry, a radar station on the ground is collecting measurements of this target. A schematic of this is shown in Figure (3.2). As in [65], assume that

- the forces acting on the object are gravity and atmospheric drag,
- the effects of centrifugal acceleration, Coriolis acceleration, wind, lift force, and spinning motion are ignored,
- the Earth is approximated as being locally flat, and
- all motion is with respect to a stationary, orthogonal x-y plane.

Take the state to be

$$\boldsymbol{x}_k = \begin{bmatrix} x_k & \dot{x}_k & y_k & \dot{y}_k \end{bmatrix}^T,$$

where $\dot{x}_k$ denotes the time rate of change (i.e. velocity) of x_k at time t_k (and similarly for $\dot{y}_k/y_k$). Differing from [65], errors in the local gravitational acceleration (g) and ballistic coefficient (β) of the target are treated as consider parameters using the methods described in this example. Additionally, errors in the nominal location of the radar (x_R, y_R) as well as sensor bias terms (b_1, b_2) are treated as consider parameters. Accordingly, define the consider parameter vector as

$$\boldsymbol{c}_k = \begin{bmatrix} g & \beta & x_R & y_R & b_1 & b_2 \end{bmatrix}^T .$$

Then, the time evolution of the target can be written as

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}) + \boldsymbol{w}_{k-1} ,$$

where

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}) &= \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{\psi}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}) \\ \boldsymbol{\psi}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}) &= -\frac{g}{2\beta}\rho(y_k)\sqrt{\dot{x}_k^2 + \dot{y}_k^2}\begin{bmatrix} \dot{x}_k \\ \dot{y}_k \end{bmatrix} \end{aligned}$$

and

$$\boldsymbol{A} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{B} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ 0 & \Delta t \end{bmatrix} ,$$

where Δt is the (constant) sample rate of the filter (taken to be the rate of radar returns). The atmospheric density is taken to follow the exponential model $\rho(y_k) = c_1 \mathrm{e}^{-c_2 y_k}$, where

$$(c_1, c_2) = \begin{cases} (1.227, 1.093 \times 10^{-4}) & \text{if } y_k < 9,144 \text{ [m]} \\ (1.754, 1.491 \times 10^{-4}) & \text{if } y_k \geq 9,144 \text{ [m]} \end{cases} .$$

The stochastic excitation in the state $\boldsymbol{w}_{k-1}$ is zero-mean, white, and taken to have covariance

$$\boldsymbol{P}_{ww,k-1} = q \cdot \text{blkdiag}\{\boldsymbol{\theta}, \boldsymbol{\theta}\}\,,$$

where

$$\boldsymbol{\theta} = \begin{bmatrix} \frac{1}{3}\Delta t^3 & \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^2 & \Delta t \end{bmatrix}$$

and q is some constant scale factor.

The consider parameters are taken to have identity dynamics with additive noise (i.e. $\boldsymbol{c}_k = \boldsymbol{c}_{k-1} + \boldsymbol{u}_{k-1}$) with zero-mean, white, noise covariance according to

$$\boldsymbol{P}_{uu,k-1} = 10^{-4} \cdot \boldsymbol{I}_{n_c \times n_c}\,.$$

It is noted that the assumption of constant ballistic coefficient is quite good for this example, since it is expected to remain nearly constant at hypersonic and high supersonic speeds [65]. In this case, the vehicle velocity only tends to approach a Mach number of 1 toward the very end of the trajectory. The inclusion of a non-zero $\boldsymbol{u}_{k-1}$ in this example is to "nudge" the terms around a small amount in the true simulation to stress the filter's ability to quantify the associated parameter statistics. Additionally, this small motion may help further model the chaotic nature of a ballistic re-entry and its associated radar observations.

The radar observations are taken to be processed range r_k and elevation ϵ_k, with uncorrupted values given by

$$\begin{aligned} r_k &= \sqrt{x_k^2 + y_k^2} \\ \epsilon_k &= \tan^{-1}(y_k/x_k)\,. \end{aligned}$$

This processing transforms the range and elevation in to Cartesian coordinates according to

$$x_k - x_R = r_k \cos(\epsilon_k)$$

$$y_k - y_R = r_k \sin(\epsilon_k)\,.$$

Then, the observation model can be written in the linear form (from [65])

$$\boldsymbol{z}_k = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_k - x_R \\ y_k - y_R \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \boldsymbol{v}_k\,.$$

This shows that only the terms $-x_R + b_1$ and $-y_R + b_2$ are observable, so estimating all four of these quantities independently should prove problematic (to a standard EKF implementation, for example). It is noted that this processing to produce a linear form is not required by the presented methods (as nonlinearities in the measurement function can be treated using either the linearization- or quadrature-based techniques presented in this section; indeed, this example examines both). This model is adopted here to be consistent with the source references.

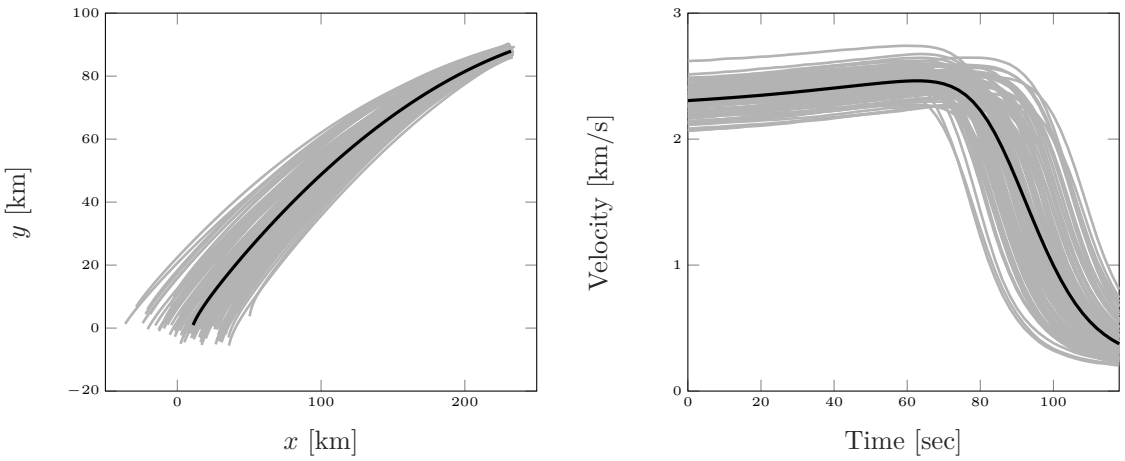
The measurement noise $\boldsymbol{v}_k$ is taken to be zero-mean and white with covariance

$$\boldsymbol{P}_{vv,k} = \begin{bmatrix} \sigma^2_{z_1,k} & \sigma_{(z_1,z_2),k} \\ \sigma_{(z_1,z_2),k} & \sigma^2_{z_2,k} \end{bmatrix}\,,$$

where

$$\begin{aligned} \sigma^2_{z_1,k} &= \sigma_r^2 \cos^2(\epsilon_k) + r_k^2 \sigma_\epsilon^2 \sin^2(\epsilon_k) \\ \sigma^2_{z_2,k} &= \sigma_r^2 \sin^2(\epsilon_k) + r_k^2 \sigma_\epsilon^2 \cos^2(\epsilon_k) \\ \sigma_{(z_1,z_2),k} &= (\sigma_r^2 - r_k^2 \sigma_\epsilon^2) \sin(\epsilon_k) \cos(\epsilon_k)\,. \end{aligned}$$

A Monte Carlo simulation of 100 trials is presented to validate the proposed approaches, as well as to demonstrate the advantages of consider filtering with square-root factors of covariance. Each Monte Carlo trial resamples the true initial state vector, true initial consider parameter vector, dynamics noise, and measurement noise. The mean and sample trajectories and velocity magnitudes are shown in Figures (3.3a) and (3.3b). In Figure (3.3b), it can be seen that at approximately 80 seconds, the drag due to the atmo-



(a) Position (b) Velocity Magnitude

Figure 3.3. Mean position and magnitude of velocity for the ballistic target (black) and all Monte Carlo sample velocity magnitudes (gray) plotted versus time.

spheric density starts to powerfully decelerate the target. Therefore, appropriate statistical quantification of the ballistic coefficient is expected to be of vital importance to estimation performance.

Radar data are collected every $\Delta t = 2$ seconds with errors according to $\sigma_r = 100$ m and $\sigma_\epsilon = 1^\circ$. The collected radar data is processing using a

- linearization-based filter that estimates, rather than considers, the vector $\boldsymbol{c}_k$ (denoted EKF),
- the described square-root linearization-based consider filter (denoted C-SREKF), and
- and the described square-root quadrature-based consider filter that utilizes the unscented transform [28, 29] (denoted C-SRUKF).[5]

The EKF is compared to the C-SREKF and C-SRUKF to illuminate the advantages afforded by an implementation that both considers uncertain parameters (vs. estimating them) and quantifies any relevant statistics with square-root factors of covariance (vs. full covariance matrices).

[5]The C-SRUKF utilizes the scaled unscented transform of [53] with parameters $\alpha = 10^{-3}$, $\beta = 2$, $\kappa = 3 - n$, where n is the dimension of a sigma point.

All three filters are initialized with the same mean

$$\boldsymbol{m}_{x,0} = \begin{bmatrix} 232,000 & \text{m} \\ v_0 \cos\gamma_0 & \text{m/s} \\ 88,000 & \text{m} \\ v_0 \sin\gamma_0 & \text{m/s} \end{bmatrix},$$

with flight path angle (with respect to the x-axis) of $\gamma_0 = 190^\circ$ and velocity magnitude of $v_0 = 2,290$ m/s, and covariance $\boldsymbol{P}_{xx,0} = \text{diag}\{1000^2, 100^2, 1000^2, 100^2\}$ for the EKF and square-root factor $\boldsymbol{S}_{xx,0} = \text{diag}\{1000, 100, 1000, 100\}$ for the C-SREKF and C-SRUKF. Additionally, the filters are initialized with the mean parameter vector

$$\boldsymbol{m}_{c,0} = \begin{bmatrix} 9.81 & \text{m/s}^2 \\ m_\beta & \text{kg/(m-s}^3\text{)} \\ \boldsymbol{0}_{4\times1} & \text{m} \end{bmatrix}$$

and covariance $\boldsymbol{P}_{cc,0} = \text{diag}\{0.01^2, \sigma_\beta^2, 25^2, 10^2, 2^2, 2^2\}$ for the EKF and square-root factor $\boldsymbol{S}_{cc,0} = \text{diag}\{0.01, \sigma_\beta, 25, 10, 2, 2\}$ for the C-SREKF and C-SRUKF. The terms m_β and σ_β^2 are the mean and variance of the uniform density $U(\beta_L, \beta_U)$ with $\beta_L = 10,000$ kg/(m-s^3) and $\beta_U = 63,000$ kg/(m-s^3). This modeling decision is inspired by the work presented by Ristic et al. [66]. In the referenced paper, the ballistic coefficient follows a beta distribution parameterized between lower and upper bounds, based on the possible cross sections of candidate objects. In this work, a uniform density (between these same bounds) is selected instead, but the idea is the same: bound ballistic coefficient by physically realizable values with an appropriate distribution.

In the following analysis, the mean error of some quantity a is denoted e_a, and the standard deviation of that error is denoted σ_a. Both the mean error and standard deviation are computed according to the errors produced by each filter over the collection of Monte Carlo trials. Random samples for the true initial state, consider parameters, etc. are drawn from Gaussian distributions as $\boldsymbol{a} \sim p_g(\boldsymbol{a}\ ;\ \boldsymbol{m}_a, \boldsymbol{P}_{aa})$, where the term $\boldsymbol{a}$ is substituted for the quantity of interest (e.g. $\boldsymbol{x}_k$, $\boldsymbol{c}_k$, etc.), and $p_g(\boldsymbol{a}\ ;\ \boldsymbol{m}_a, \boldsymbol{P}_{aa})$ denotes a Gaussian distribution in $\boldsymbol{a}$ with mean $\boldsymbol{m}_a$ and covariance $\boldsymbol{P}_{aa}$. The mean error for the position and velocity terms can be found in Figures (3.4) and (3.5), respectively. Additionally, the standard deviations

of the position and velocity terms can be found in Figures (3.6) and (3.7). Note that, while not shown to keep figures from becoming overly cluttered, single-run filter performances for the consider filters were found to be consistent with the Monte Carlo statistics.

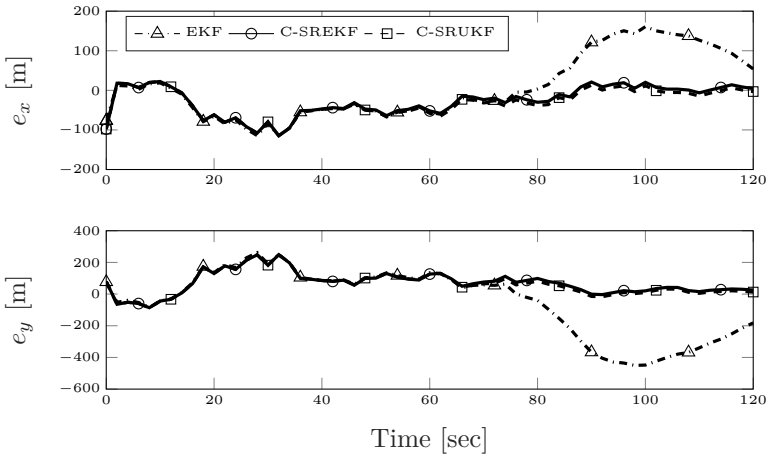


Figure 3.4. Monte Carlo mean errors for the position states.

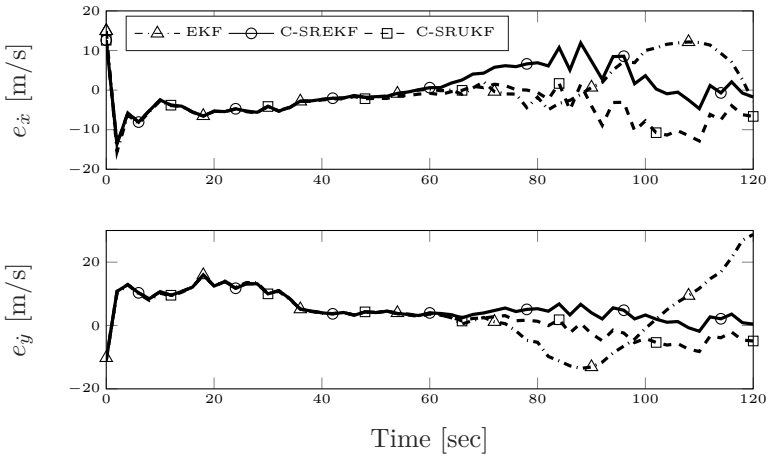


Figure 3.5. Monte Carlo mean errors for the velocity states.

First, of key importance to the following discussion is that the EKF failed in 2 of the 100 trials.[6] Accordingly, the presented statistics are averaged over the remaining 98 trials, justifying the small differences in mean error between the filters at $t_0 = 0$ seconds

[6]In this context, a failure refers any time the filter fails to maintain a SPD covariance matrix at any point in the recursion.

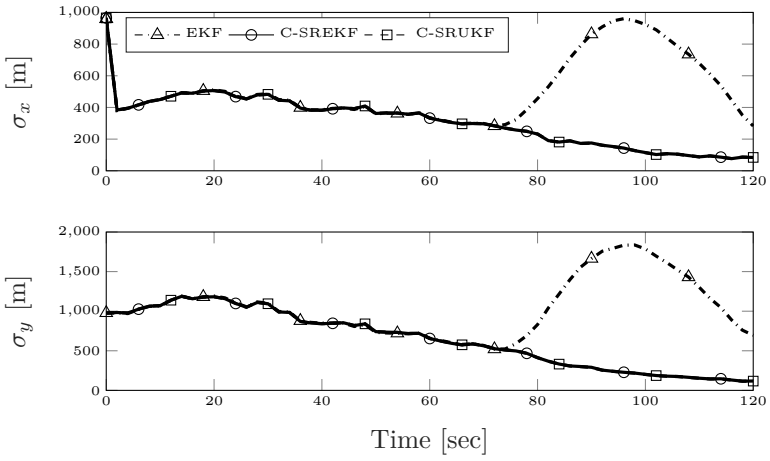


Figure 3.6. Monte Carlo standard deviations for the position states.

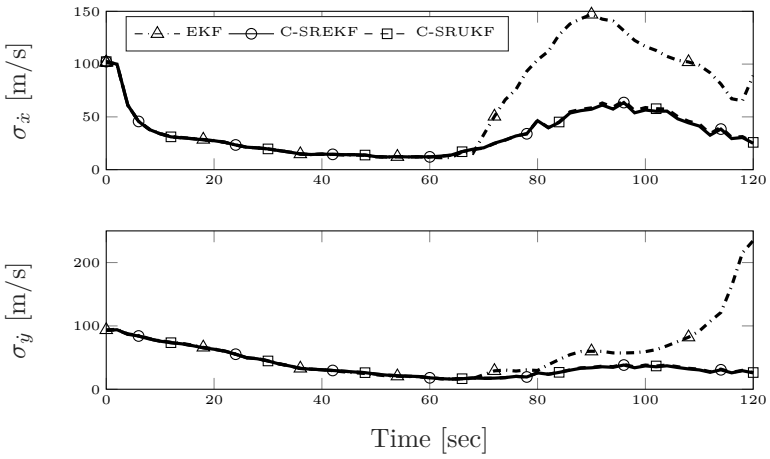


Figure 3.7. Monte Carlo standard deviations for the velocity states.

of the filter run (such as in Figure (3.4)). Comparison to the other filters, both of which exhibited no failures over the collection of trials, demonstrates that the EKF has trouble appropriately quantifying the errors induced by incorrect parameter estimates.[7]

Second, the errors shown in Figures (3.4) and (3.5) indicate that the EKF is outperformed by the C-SREKF and C-SRUKF, and that the consider filters better quantify errors in the high-acceleration region of the trajectory (post 80 seconds). The EKF demonstrates instability, and even begins to diverge toward the end of the simulation. These results are corroborated by the standard deviations shown in Figures (3.6) and (3.7), where the

[7]This conclusion is consistent with the study of Julier in [29] on a related problem.

C-SREKF and C-SRUKF perform nearly identically and the EKF can be seen to perform worse, even apparently diverging toward the end of the simulation (particularly apparent in Figure (3.7)).

Third, while the error plots in Figures (3.4) and (3.5) seem to indicate that the C-SREKF performs slightly better than the C-SRUKF, their performances are so similar (and since the standard deviations in Figures (3.6) and (3.7) are almost identical) that no strong conclusions about one performing better than the other can be claimed in this case.

Finally, to demonstrate that a single run of the filters (i.e. one Monte Carlo trial) well-represents the statistics seen in Figures (3.4)–(3.7), a measure of consistency is presented. Using a single trial as reference, the term $|\boldsymbol{P}_{\mathrm{MC},k}^{-1}\boldsymbol{P}_{xx,k} - \boldsymbol{I}_{4\times 4}|$ is computed, where $\boldsymbol{P}_{\mathrm{MC},k}$ is the error covariance of $\boldsymbol{x}_k$ computed over all the Monte Carlo trials, $\boldsymbol{P}_{xx,k}$ is the estimated covariance matrix of $\boldsymbol{x}_k$ at t_k from the single filtering run, and $|\cdot|$ denotes the determinant. This term quantifies a "difference" of sorts from the Monte Carlo covariance and the covariance of a single filter run, a number that should be small if a filter is appropriately quantifying the underlying statistics. The results of this analysis can be seen in Figure (3.8), where the reference trial is trial 74, a number chosen at random between 1 and 100 with respect to a uniform distribution. This figure indicates the expected result: the C-SREKF and C-SRUKF have very small differences from their associated Monte Carlo statistics, and the EKF demonstrates much larger differences when the ballistic target begins its strong deceleration. This indicates that, consistent with previous analyses, the EKF poorly quantifies its error statistics.

3.3. BAYESIAN CONSIDER FILTERS

A Roadmap. The following developments are motivated by early seminal works in the topic of sequential filtering, and a "roadmap" illustrating the motivation and process for this discussion is outlined in Figure 3.9, the contents of which are summarily described presently. Some of the initialisms employed here will not appear later, but they serve as a convenient vehicle for compact exposition.

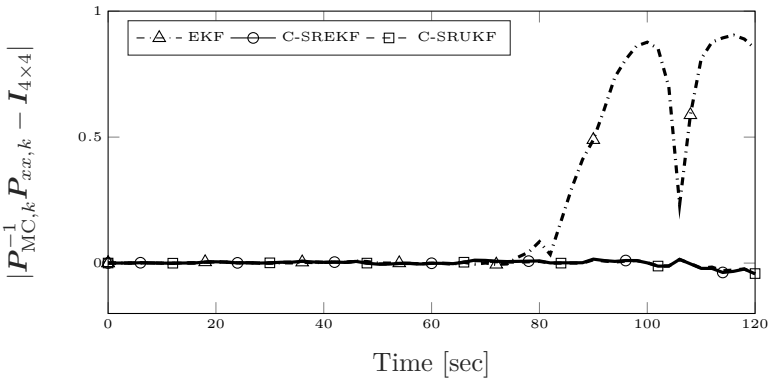


Figure 3.8. A "consistency metric" comparing the error covariance of a single filtering run with its associated Monte Carlo error statistics.

With the use of MMSE principles that estimate mean and covariance (denoted $\boldsymbol{m}$ & $\boldsymbol{P}$), Kalman produced his now-ubiquitous filter (denoted KF) [1, 2], and briefly thereafter, Schmidt introduced the concept of consider filtering under the same framework (denoted CKF) [18, 55]. A short time later, Ho demonstrated that an algorithmic equivalent to Kalman's filter (denoted BKF) can be obtained using Bayes' rule if one assumes the associated uncertainties are Gaussian-distributed [67], where the algorithmic equivalence is denoted as the double-sided black arrow connecting KF and BKF. Somewhat later, Sorenson showed that a much more general nonlinear, non-Gaussian estimator can be obtained via the use of GM parameterizations (denoted GMKF) [62, 68]. This was a direct result of Ho's developments, thus the single-sided unfilled arrow from BKF to GMKF. It turns out that that most GM-driven multitarget filters are a direct consequence of the GMKF, as plainly seen in [69, 70, 71, 72, 73]. This dissertation ultimately seeks to produce multitarget filters that employ consider parameters, and, therefore, these early developments are emulated, but now with consider filters in mind. The required tools were previously unavailable, denoted by the use of broken lines in the figure and are innovations of this dissertation. First, the consider filter is derived using Bayes' rule in Section 3.3.2 (denoted in the figure as BCKF), and Section 3.3.3 demonstrates that this directly implies the GM consider filter (denoted in the figure as GMCKF). Then, these results are ultimately leveraged in Section 4 to derive multitarget consider filters.

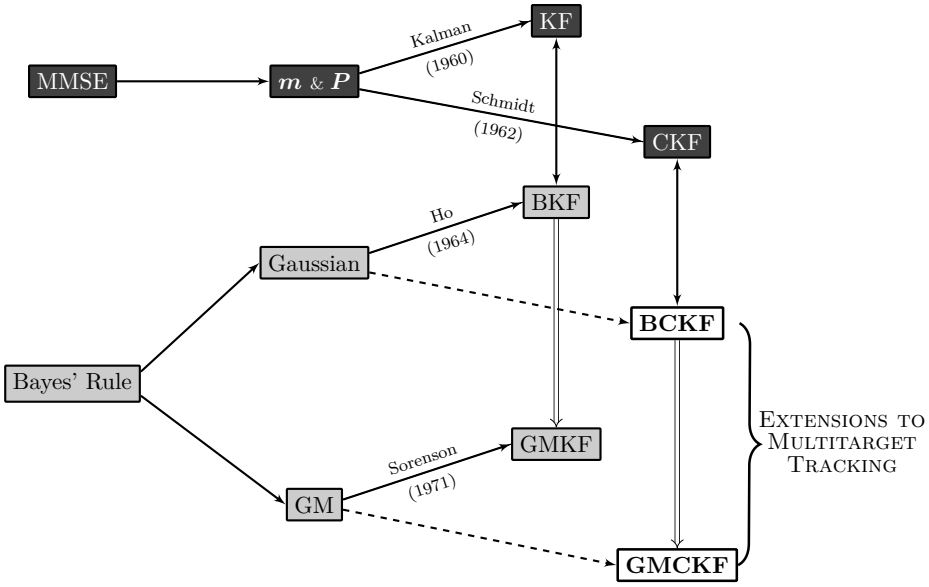


Figure 3.9. Roadmap to obtaining consider filters using Bayes' rule, ultimately producing multitarget consider filters.

A reader will notice that the following is presented in terms of mean and covariance, rather than mean and square-root factor as discussed at length in the previous section, Section 3.2. This is for convenience as it somewhat compacts the discussion, but it should be clear at this point how to take the following results and produce square-root consider formulations of these filters.

Remark 3.4. *As discussed in the introduction to this dissertation, it might seem pedantic to distinguish MMSE and Bayes' rule as different paradigms. After all (using the terminology of Figure 3.9), aren't the KF and the BKF algorithmically equivalent, and don't both filters estimate mean and covariance? The key here is in the difference of underlying assumptions required to produce the resulting filters. The reader is asked to refer back to Section 1 for more discussion on this point, but this distinction is key to illustrating the inspiration and direction of the approach.*

3.3.1. Preliminaries. The following describes some necessary mathematical details required to formulate the desired filters. In particular, the connection between Bayes' rule and consider filters is explored, the necessary dynamical and observational system modeling is described, and then new properties of Gaussian densities required to produce the results of later sections are presented.

3.3.1.1. Bayesian inference under the influence of consider parameters. In discrete-time, Bayesian filtering problems concerned with estimating the states of a single target, the state probability density function (pdf) at a previous time is propagated forward in time to the epoch of an incoming measurement of that target.[8] The resulting density is called the *a priori* pdf. Then, the new measurement information is fused with the *a priori* pdf to produce the target's *a posteriori* density conditioned on that information. For some arbitrary state vector at k denoted $\boldsymbol{x}_k$, recursively processing data as it is acquired, with some measurement history up to and including $\boldsymbol{z}_k$ denoted as $\boldsymbol{Z}_{1:k} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k\}$, can be performed using the Chapman-Kolmogorov equation, given by

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k-1}) = \int f(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{1:k-1}) \mathrm{d}\boldsymbol{x}_{k-1} \,, \tag{3.36}$$

and Bayes' rule, given by

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k}) = \frac{1}{\eta_k} g(\boldsymbol{z}_k|\boldsymbol{x}_k) p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k-1}) \,, \tag{3.37}$$

where $f(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ is the single-target transition density, $g(\boldsymbol{z}_k|\boldsymbol{x}_k)$ is the measurement likelihood function, and η_k is an appropriate normalization constant such that the posterior density is indeed a proper density. The term $p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k})$ denotes the pdf of $\boldsymbol{x}_k$ conditioned on all data up to and including $\boldsymbol{z}_k$. In Eqs. (3.36) and (3.37), $p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k-1})$ represents the *a priori* distribution and $p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k})$ represents the *a posteriori* distribution. In the presence of uncertain model parameters at k, given by $\boldsymbol{c}_k$, the recursion formed by the Chapman-Kolmogorov equation and Bayes' rule works with the joint density of the state

[8] The terminology "target" is used to be consistent with tracking literature, but, in reality, this estimated quantity can pertain to any state vector of interest, be it corresponding to a target, a vehicle, etc.

and parameters to yield

$$p(\boldsymbol{x}_k, \boldsymbol{c}_k|\boldsymbol{Z}_{1:k-1}) = \int f(\boldsymbol{x}_k, \boldsymbol{c}_k|\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}) p(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}|\boldsymbol{Z}_{1:k-1}) \mathrm{d}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}) \tag{3.38}$$

and

$$p(\boldsymbol{x}_k, \boldsymbol{c}_k|\boldsymbol{Z}_{1:k}) = \frac{1}{\eta_k} g(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{c}_k) p(\boldsymbol{x}_k, \boldsymbol{c}_k|\boldsymbol{Z}_{1:k-1}) \,, \tag{3.39}$$

If the desire is to *estimate* the values of both the state and the parameters, Eqs. (3.38) and (3.39) precisely describe the stochastic evolution of the system. If, however, the goal is to estimate only the values of the state and *consider* the effects of the uncertainties in the system parameters, the parameters can be marginalized out via

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k-1}) = \iint f(\boldsymbol{x}_k, \boldsymbol{c}_k|\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}) p(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}|\boldsymbol{Z}_{1:k-1}) \mathrm{d}(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}) \mathrm{d}\boldsymbol{c}_k \tag{3.40}$$

and

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k}) = \int \frac{1}{\eta_k} g(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{c}_k) p(\boldsymbol{x}_k, \boldsymbol{c}_k|\boldsymbol{Z}_{1:k-1}) \mathrm{d}\boldsymbol{c}_k \,. \tag{3.41}$$

Equations (3.40) and (3.41) describe the Bayesian filtering recursion that produces the conditional state density as influenced by the uncertainties in the system parameters without producing information of the parameter values themselves. This is where the key to deriving a Bayesian analog of MMSE consider filtering is discovered: *marginalization of the consider parameters*. It turns out that this approach, using further assumptions detailed in the following discussion, produces the desired algorithmic equivalence to traditional, MMSE consider filtering. Before these assumptions are imposed, however, important modeling decisions must be imposed to enable the resulting closed-form results.

3.3.1.2. System modeling. Some of the following assumptions, such as taking the consider parameter density to be Gaussian, are unneeded to produce closed, practical recursions, but greatly aid in ease of exposition. Extending these results to the case of non-Gaussian consider parameter densities, using a Gaussian mixture pdf representa-

tion, follows handily but congests discussion and is therefore omitted. Note that, where possible, all notation and modeling system designs follow that of Section 3.1, such as the dynamical models of Eqs. (3.1) and observational model Eq. (3.2) on pp. 59. However, all of this discussion is presented for linear systems only. Using the same methods described in Section 3.1 or Section 3.2 permit linearization- or quadrature-based full covariance or square-root implementations, respectively. A reader equipped with those sections will find that they naturally extend to these results.

Assume that the n_x-dimensional vector $\boldsymbol{x}_{k-1}$ evolves stochastically according to the Gaussian transition density given by

$$f(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}) = p_g(\boldsymbol{x}_k\,;\, \boldsymbol{F}_{x,k-1}\boldsymbol{x}_{k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{c}_{k-1}, \boldsymbol{F}_{w,k-1}\boldsymbol{P}_{ww,k-1}\boldsymbol{F}_{w,k-1}^T)\,, \tag{3.42}$$

where $\boldsymbol{F}_{x,k-1}$ is the state transition matrix, $\boldsymbol{c}_{k-1}$ is the n_c-dimensional vector of consider parameters, $\boldsymbol{F}_{c,k-1}$ is an $n_x \times n_c$ matrix that maps the consider parameter vector into the evolution of the state, and $\boldsymbol{P}_{ww,k-1}$ is the process noise covariance. The vector $\boldsymbol{c}_{k-1}$ is taken to evolve according to the Gaussian transition density

$$f(\boldsymbol{c}_k|\boldsymbol{c}_{k-1}) = p_g(\boldsymbol{c}_k\,;\, \boldsymbol{G}_{c,k-1}\boldsymbol{c}_{k-1}, \boldsymbol{G}_{u,k-1}\boldsymbol{P}_{uu,k-1}\boldsymbol{G}_{u,k-1}^T)\,, \tag{3.43}$$

where $\boldsymbol{G}_{c,k-1}$ is the transition matrix for $\boldsymbol{c}_{k-1}$ and $\boldsymbol{P}_{uu,k-1}$ is the covariance matrix of the process noise that is a stochastic input to the temporal evolution of $\boldsymbol{c}_{k-1}$. Observations are taken to be governed by the Gaussian likelihood

$$g(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{c}_k) = p_g(\boldsymbol{z}_k\,;\, \boldsymbol{H}_{x,k}\boldsymbol{x}_k + \boldsymbol{H}_{c,k}\boldsymbol{c}_k, \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}_{v,k}^T)\,, \tag{3.44}$$

where $\boldsymbol{z}_k$ is an n_z-dimensional observation at k, $\boldsymbol{H}_{x,k}$ is an $n_z \times n_x$ matrix that maps the state into the measurement space, $\boldsymbol{H}_{c,k}$ is an $n_z \times n_c$ matrix that maps the consider parameters into the observation space, and $\boldsymbol{P}_{vv,k}$ is the covariance matrix of the measurement noise.

Remark 3.5. *Note that, under these assumptions, Eqs. (3.42)–(3.44) exactly correspond to the dynamical models of Eqs. (3.1) and observational model Eq. (3.2) under the constraint that the system is linear and the noises are Gaussian.*

Remark 3.6. *While the dynamical and observational systems have contributions from the same consider parameter vector* $\boldsymbol{c}_k$*, the matrices* $\boldsymbol{F}_{c,k-1}$ *and* $\boldsymbol{H}_{c,k}$ *map elements from this vector appropriately so that they need not be the same elements in* $\boldsymbol{c}_k$ *in each case. As an example, consider the one-dimensional state-space model*

$$\begin{aligned} x_k &= F_{x,k-1}x_{k-1} + \xi_{k-1} + w_{k-1} \\ z_k &= H_{x,k}x_k + \zeta_k + v_k \,, \end{aligned}$$

with uncertain parameters ξ_{k-1} *and* ζ_k *and noises* w_{k-1} *and* v_k*. Then, the consider parameter vector is given by*

$$\boldsymbol{c}_k = \begin{bmatrix} \xi_k & \zeta_k \end{bmatrix}^T ,$$

and the matrices $\boldsymbol{F}_{c,k-1}$ *and* $\boldsymbol{H}_{c,k}$ *are*

$$\begin{aligned} \boldsymbol{F}_{c,k-1} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \\ \boldsymbol{H}_{c,k} &= \begin{bmatrix} 0 & 1 \end{bmatrix} . \end{aligned}$$

Then, the state-space model equations can be written as

$$\begin{aligned} x_k &= F_{x,k-1}x_{k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{c}_{k-1} + w_{k-1} \\ z_k &= H_{x,k}x_k + \boldsymbol{H}_{c,k}\boldsymbol{c}_k + v_k \,, \end{aligned}$$

which are called the system process and observation process, respectively.

It is also required to describe the correlations between the state and consider parameters. As such, let the matrix $\boldsymbol{P}_{xc,k}$, which captures correlations between the state and consider parameter vector, be defined as

$$\boldsymbol{P}_{xc,k} = \mathbb{E}\left\{ (\boldsymbol{x}_k - \boldsymbol{m}_{x,k})(\boldsymbol{c}_k - \boldsymbol{m}_{c,k})^T \right\} \in \mathbb{R}^{n_x \times n_c} .$$

If no *a priori* knowledge of these correlations is available, it would be reasonable to set this term to an array of zeros.

3.3.1.3. Necessary properties of Gaussian densities. In order to produce the results of the following section, some useful properties regarding Gaussian densities are required. Lemmas 3.1 and 3.2 are widely known results produced by Ho's work in [67], and are thus termed "Ho's equations", as integral and product forms, respectively. Lemmas 3.3 and 3.4 are new generalizations of these expressions shown here and are utilized to produce useful results for consider filters.

Lemma 3.1 (Ho's Equation: Integral Form)**.** *Given terms of appropriate dimension and provided that* $\boldsymbol{Q}$ *and* $\boldsymbol{P}$ *are positive definite, it can be shown that*

$$\int p_g(\boldsymbol{x}\,;\,\boldsymbol{F\xi},\boldsymbol{Q})p_g(\boldsymbol{\xi}\,;\,\boldsymbol{m},\boldsymbol{P})\mathrm{d}\boldsymbol{\xi} = p_g(\boldsymbol{x}\,;\,\boldsymbol{Fm},\boldsymbol{FPF}^T+\boldsymbol{Q})\,.$$

Lemma 3.2 (Ho's Equation: Product Form)**.** *Given terms of appropriate dimension and provided that* $\boldsymbol{R}$ *and* $\boldsymbol{P}$ *are positive definite,*

$$p_g(\boldsymbol{z}\,;\,\boldsymbol{Hx},\boldsymbol{R})p_g(\boldsymbol{x}\,;\,\boldsymbol{m},\boldsymbol{P}) = q(\boldsymbol{z})p_g(\boldsymbol{x}\,;\,\boldsymbol{\mu},\boldsymbol{\Sigma})\,,$$

where

$$\begin{aligned} q(\boldsymbol{z}) &= p_g(\boldsymbol{z}\,;\,\boldsymbol{Hm},\boldsymbol{HPH}^T+\boldsymbol{R}) \\ \boldsymbol{\mu} &= \boldsymbol{m}+\boldsymbol{K}(\boldsymbol{z}-\boldsymbol{Hm}) \\ \boldsymbol{\Sigma} &= (\boldsymbol{I}-\boldsymbol{KH})\boldsymbol{P} \\ \boldsymbol{K} &= \boldsymbol{PH}^T(\boldsymbol{HPH}^T+\boldsymbol{R})^{-1}\,. \end{aligned}$$

Lemma 3.3 (A Generalization of Ho's Equation: Integral Form)**.** *Given vectors* $\boldsymbol{x}\in\mathbb{R}^n$, $\boldsymbol{b}\in\mathbb{R}^\ell$, *and* $\boldsymbol{\xi}\in\mathbb{R}^n$ *and matrices* $\boldsymbol{F}\in\mathbb{R}^{n\times n}$, $\boldsymbol{G}\in\mathbb{R}^{n\times\ell}$, $\boldsymbol{M}\in\mathbb{R}^{\ell\times\ell}$, $\boldsymbol{P}\in\mathbb{R}^{n\times n}$, $\boldsymbol{Q}\in\mathbb{R}^{n\times n}$, $\boldsymbol{\Omega}\in\mathbb{R}^{\ell\times\ell}$, $\boldsymbol{L}\in\mathbb{R}^{n\times\ell}$, *and* $\boldsymbol{B}\in\mathbb{R}^{\ell\times\ell}$, *where* $\boldsymbol{Q}$, $\boldsymbol{\Omega}$, $\boldsymbol{P}$, *and* $\boldsymbol{B}$ *are symmetric, positive*

definite matrices,

$$\int p_g\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{b}\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{F} & \boldsymbol{G}\\ \boldsymbol{0} & \boldsymbol{M}\end{bmatrix}\begin{bmatrix}\boldsymbol{\xi}\\ \boldsymbol{b}\end{bmatrix},\begin{bmatrix}\boldsymbol{Q} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{\Omega}\end{bmatrix}\right)\times p_g\left(\begin{bmatrix}\boldsymbol{\xi}\\ \boldsymbol{b}\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{m}\\ \boldsymbol{p}\end{bmatrix},\begin{bmatrix}\boldsymbol{P} & \boldsymbol{L}\\ \boldsymbol{L}^T & \boldsymbol{B}\end{bmatrix}\right)\mathrm{d}\begin{bmatrix}\boldsymbol{\xi}\\ \boldsymbol{b}\end{bmatrix}$$
$$= p_g\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{b}\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\eta}\end{bmatrix},\begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{\Pi}\\ \boldsymbol{\Pi}^T & \boldsymbol{\Xi}\end{bmatrix}\right)$$

where

$$\begin{aligned}\boldsymbol{\mu} &= \boldsymbol{F}\boldsymbol{m} + \boldsymbol{G}\boldsymbol{p}\\ \boldsymbol{\eta} &= \boldsymbol{M}\boldsymbol{p}\\ \boldsymbol{\Sigma} &= \boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^T + \boldsymbol{G}\boldsymbol{B}\boldsymbol{G}^T + \boldsymbol{G}\boldsymbol{L}^T\boldsymbol{F}^T + \boldsymbol{F}\boldsymbol{L}\boldsymbol{G}^T + \boldsymbol{Q}\\ \boldsymbol{\Pi} &= \boldsymbol{F}\boldsymbol{L}\boldsymbol{M}^T + \boldsymbol{G}\boldsymbol{B}\boldsymbol{M}^T\\ \boldsymbol{\Xi} &= \boldsymbol{M}\boldsymbol{B}\boldsymbol{M}^T + \boldsymbol{\Omega}\,.\end{aligned}$$

Proof is given in Appendix C.5.

Lemma 3.4 (A Generalization of Ho's Equation: Marginalized Product Form)**.** *Given vectors* $\boldsymbol{x} \in \mathbb{R}^n$*,* $\boldsymbol{b} \in \mathbb{R}^\ell$*, and* $\boldsymbol{z} \in \mathbb{R}^m$ *and matrices* $\boldsymbol{H} \in \mathbb{R}^{m\times n}$*,* $\boldsymbol{J} \in \mathbb{R}^{m\times \ell}$*,* $\boldsymbol{P} \in \mathbb{R}^{n\times n}$*,* $\boldsymbol{L} \in \mathbb{R}^{n\times \ell}$*, and* $\boldsymbol{B} \in \mathbb{R}^{\ell\times\ell}$*, where* $\boldsymbol{R}$*,* $\boldsymbol{P}$*, and* $\boldsymbol{B}$ *are symmetric, positive definite matrices,*

$$\int p_g(\boldsymbol{z}\,;\,\boldsymbol{H}\boldsymbol{x}+\boldsymbol{J}\boldsymbol{b},\boldsymbol{R})\,p_g\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{b}\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{m}\\ \boldsymbol{p}\end{bmatrix},\begin{bmatrix}\boldsymbol{P} & \boldsymbol{L}\\ \boldsymbol{L}^T & \boldsymbol{B}\end{bmatrix}\right)\mathrm{d}\boldsymbol{b} = q(\boldsymbol{z})\,p_g(\boldsymbol{x}\,;\,\boldsymbol{\mu},\boldsymbol{\Sigma})$$

where

$$\begin{aligned}q(\boldsymbol{z}) &= p_g(\boldsymbol{z}\,;\,\boldsymbol{H}\boldsymbol{m}+\boldsymbol{J}\boldsymbol{p},\boldsymbol{W})\\ \boldsymbol{\mu} &= \boldsymbol{m} + \boldsymbol{K}(\boldsymbol{z}-\boldsymbol{H}\boldsymbol{m}-\boldsymbol{J}\boldsymbol{p})\\ \boldsymbol{\Sigma} &= (\boldsymbol{I}-\boldsymbol{K}\boldsymbol{H})\boldsymbol{P} - \boldsymbol{K}\boldsymbol{J}\boldsymbol{L}^T\\ \boldsymbol{K} &= \boldsymbol{C}\boldsymbol{W}^{-1}\\ \boldsymbol{C} &= \boldsymbol{P}\boldsymbol{H}^T + \boldsymbol{L}\boldsymbol{J}^T\\ \boldsymbol{W} &= \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^T + \boldsymbol{J}\boldsymbol{B}\boldsymbol{J}^T + \boldsymbol{H}\boldsymbol{L}\boldsymbol{J}^T + \boldsymbol{J}\boldsymbol{L}^T\boldsymbol{H}^T + \boldsymbol{R}\,.\end{aligned}$$

Proof is given in Appendix C.6.

The following discussion relies upon the use of the widely adored Gaussian pdf. Conveniently, Gaussian densities are fully parameterized solely by their first two statistical moments, mean and covariance, and thus are typically expressed in the form

$$p_g(\boldsymbol{x}_k\,;\,\boldsymbol{m}_{x,k},\boldsymbol{P}_{xx,k}) = |2\pi\boldsymbol{P}_{xx,k}|^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}(\boldsymbol{x}_k-\boldsymbol{m}_{x,k})^T\boldsymbol{P}_{xx,k}^{-1}(\boldsymbol{x}_k-\boldsymbol{m}_{x,k})\right\},$$

with mean $\boldsymbol{m}_{x,k}$ and covariance $\boldsymbol{P}_{xx,k}$ and where $|\cdot|$ denotes the determinant. However, this dissertation has described formulations that rely upon mean and square-root factor, $\boldsymbol{m}_{x,k}$ and $\boldsymbol{S}_{xx,k}$, rather than mean and covariance. Therefore, if $\boldsymbol{S}_{xx,k}\boldsymbol{S}_{xx,k}^T = \boldsymbol{P}_{xx,k}$, it is useful to note that Gaussian densities can also be conveniently represented in terms of square-root factors as

$$p_g(\boldsymbol{x}_k\,;\,\boldsymbol{m}_{x,k},\boldsymbol{S}_{xx,k}) = \text{abs}\left\{\left|\sqrt{2\pi}\;\boldsymbol{S}_{xx,k}\right|\right\}^{-1}\exp\left\{-\frac{1}{2}\left|\left|\boldsymbol{S}_{xx,k}^{-1}(\boldsymbol{x}_k-\boldsymbol{m}_{x,k})\right|\right|_2^2\right\},$$

where $||\cdot||_2^2$ denotes the squared 2-norm and $\text{abs}\{\cdot\}$ denotes the absolute value.

3.3.2. Gaussian State Densities. Ho demonstrated that an algorithmic equivalent of the Kalman filter can be obtained via the Chapman-Kolmogorov equation and Bayes' rule when the transition, likelihood, and conditional state densities are all Gaussian [67]. As described in Section 3.3.1.2, the following discussion makes the same assumptions regarding the transition, likelihood, and conditional state densities and also assumes that the uncertain parameters are Gaussian distributed. Then, using Lemmas 3.3 and 3.4 shown in the previous section produces an algorithmic equivalent to the linear MMSE consider filter described in Section 3.1.

3.3.2.1. Predictor. Assuming that the posterior state density at time t_{k-1} is Gaussian of the form

$$p(\boldsymbol{x}_{k-1}|\boldsymbol{Z}_{1:k-1}) = p_g(\boldsymbol{x}_{k-1}\,;\,\boldsymbol{m}_{x,k-1}^+,\boldsymbol{P}_{xx,k-1}^+)$$

and that the system and parameters are described as in Section 3.3.1.2, the posterior joint density at time t_{k-1} of the state and the parameters can be written as

$$p(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}|\boldsymbol{Z}_{1:k-1}) = p_g\left(\begin{bmatrix}\boldsymbol{x}_{k-1}\\ \boldsymbol{c}_{k-1}\end{bmatrix}\ ;\ \begin{bmatrix}\boldsymbol{m}^+_{x,k-1}\\ \boldsymbol{m}^+_{c,k-1}\end{bmatrix}, \begin{bmatrix}\boldsymbol{P}^+_{xx,k-1} & \boldsymbol{P}^+_{xc,k-1}\\ (\boldsymbol{P}^+_{xc,k-1})^T & \boldsymbol{P}^+_{cc,k-1}\end{bmatrix}\right), \tag{3.45}$$

where one should recall that the matrix $\boldsymbol{P}^+_{xc,k-1}$ captures correlations between the posterior state and parameters at $k-1$. Then, the consider filter prediction equations for the joint density of the state and parameters are given as

$$p(\boldsymbol{x}_k, \boldsymbol{c}_k|\boldsymbol{Z}_{1:k-1}) = p_g\left(\begin{bmatrix}\boldsymbol{x}_k\\ \boldsymbol{c}_k\end{bmatrix}\ ;\ \begin{bmatrix}\boldsymbol{m}^-_{x,k}\\ \boldsymbol{m}^-_{c,k}\end{bmatrix}, \begin{bmatrix}\boldsymbol{P}^-_{xx,k} & \boldsymbol{P}^-_{xc,k}\\ (\boldsymbol{P}^-_{xc,k})^T & \boldsymbol{P}^-_{cc,k}\end{bmatrix}\right), \tag{3.46}$$

where

$$\boldsymbol{m}^-_{x,k} = \boldsymbol{F}_{x,k-1}\boldsymbol{m}^+_{x,k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{m}^+_{c,k-1} \tag{3.47a}$$

$$\boldsymbol{m}^-_{c,k} = \boldsymbol{G}_{c,k-1}\boldsymbol{m}^+_{c,k-1} \tag{3.47b}$$

$$\begin{aligned}\boldsymbol{P}^-_{xx,k} = {} & \boldsymbol{F}_{x,k-1}\boldsymbol{P}^+_{xx,k-1}\boldsymbol{F}^T_{x,k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{P}^+_{cc,k-1}\boldsymbol{F}^T_{c,k-1} + \boldsymbol{F}_{w,k-1}\boldsymbol{P}_{ww,k-1}\boldsymbol{F}^T_{w,k-1}\\ & + \boldsymbol{F}_{x,k-1}\boldsymbol{P}^+_{xc,k-1}\boldsymbol{F}^T_{c,k-1} + \boldsymbol{F}_{c,k-1}(\boldsymbol{P}^+_{xc,k-1})^T\boldsymbol{F}^T_{x,k-1}\end{aligned} \tag{3.47c}$$

$$\boldsymbol{P}^-_{cc,k} = \boldsymbol{G}_{c,k-1}\boldsymbol{P}^+_{cc,k-1}\boldsymbol{G}^T_{c,k-1} + \boldsymbol{G}_{u,k-1}\boldsymbol{P}_{uu,k-1}\boldsymbol{G}^T_{u,k-1} \tag{3.47d}$$

$$\boldsymbol{P}^-_{xc,k} = \boldsymbol{F}_{x,k-1}\boldsymbol{P}^+_{xc,k-1}\boldsymbol{G}^T_{c,k-1} + \boldsymbol{F}_{x,k-1}\boldsymbol{P}^+_{cc,k-1}\boldsymbol{G}^T_{c,k-1}\,. \tag{3.47e}$$

Proof. Form the joint transition density using the product

$$f(\boldsymbol{x}_k, \boldsymbol{c}_k|\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}) = f(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1})f(\boldsymbol{c}_k|\boldsymbol{c}_{k-1})\,,$$

where $f(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1})$ and $f(\boldsymbol{c}_k|\boldsymbol{c}_{k-1})$ are given in Eqs. (3.42) and (3.43), respectively. Substituting this and Eq. (3.45) into Eq. (3.38) and applying Lemma 3.3 produces the claimed result. ■

Remark 3.7. *It is immediately apparent that these expressions are identical to the prediction equations derived under MMSE principles in Eqs. (3.12) on pp. 66 in the case of a linear system. Again, extending these results to square-root formulations and/or nonlinear systems follows from the previous developments.*

Equations (3.46) and (3.47) provide the joint prior density for the state and consider parameters at time t_k. If only the marginal state density is desired (as is the case in consider filters, where updates from the considered parameters are desired to be "discarded"), it is possible to factor the prior joint density as

$$p(\boldsymbol{x}_k, \boldsymbol{c}_k | \boldsymbol{Z}_{1:k-1}) = p(\boldsymbol{x}_k | \boldsymbol{Z}_{1:k-1}) p(\boldsymbol{c}_k | \boldsymbol{x}_k, \boldsymbol{Z}_{1:k-1}) \,;$$

then, marginalization over $\boldsymbol{c}_k$ yields only the state density $p(\boldsymbol{c}_k | \boldsymbol{Z}_{1:k-1}) = p_g(\boldsymbol{x}_k \,;\, \boldsymbol{m}^-_{x,k}, \boldsymbol{P}^-_{xx,k})$. This result can also be obtained by appropriately placing terms in Eq. (3.40) and integrating with Lemma 3.3.

Remark 3.8. *Note that in the case of stationary consider parameters (that is, $\boldsymbol{G}_{c,k-1} = \boldsymbol{I}$ and $\boldsymbol{P}_{uu,k-1} = \boldsymbol{0}_{n_u \times n_u}$), Eqs. (3.47b) and (3.47d) need not be utilized in the prediction stage. This is to say that, if the consider parameters have identity dynamics, $\boldsymbol{m}^-_{c,k} = \boldsymbol{m}^+_{c,k-1}$ and $\boldsymbol{P}^-_{cc,k} = \boldsymbol{P}^+_{cc,k-1}$, $\forall\, k > 0$.*

3.3.2.2. Corrector. Assume now that a measurement $\boldsymbol{z}_k$, with likelihood function given by Eq. (3.44), is received and is to be processed. Given the joint prior density of Eq. (3.46), Eq. (3.41) (with appropriate normalization constant η_k) and Lemma 3.4 can be used to yield the marginal posterior distribution at time t_k as

$$p(\boldsymbol{x}_k | \boldsymbol{Z}_{1:k}) = p_g(\boldsymbol{x}_k \,;\, \boldsymbol{m}^+_{x,k}, \boldsymbol{P}^+_{xx,k}) \,,$$

where

$$\boldsymbol{m}^+_{x,k} = \boldsymbol{m}^-_{x,k} + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{H}_{x,k}\boldsymbol{m}^-_{x,k} - \boldsymbol{H}_{c,k}\boldsymbol{m}^-_{c,k}) \tag{3.48a}$$

$$\boldsymbol{P}^+_{xx,k} = (\boldsymbol{I}_{n_x \times n_x} - \boldsymbol{K}_{x,k}\boldsymbol{H}_{x,k})\boldsymbol{P}^-_{xx,k} - \boldsymbol{K}_{x,k}\boldsymbol{H}_{c,k}(\boldsymbol{P}^-_{xc,k})^T \tag{3.48b}$$

$$\boldsymbol{P}^+_{xc,k} = (\boldsymbol{I}_{n_x \times n_x} - \boldsymbol{K}_{x,k}\boldsymbol{H}_{x,k})\boldsymbol{P}^-_{xc,k} - \boldsymbol{K}_{x,k}\boldsymbol{H}_{c,k}\boldsymbol{P}^-_{cc,k} \tag{3.48c}$$

$$\boldsymbol{K}_{x,k} = \boldsymbol{P}_{xz,k}\boldsymbol{P}_{zz,k}^{-1} \tag{3.48d}$$

$$\boldsymbol{P}_{xz,k}^{-} = \boldsymbol{P}_{xx,k}^{-}\boldsymbol{H}_{x,k}^{T} + \boldsymbol{P}_{xc,k}^{-}\boldsymbol{H}_{c,k}^{T} \tag{3.48e}$$

$$\begin{aligned}\boldsymbol{P}_{zz,k}^{-} = {} & \boldsymbol{H}_{x,k}\boldsymbol{P}_{xx,k}^{-}\boldsymbol{H}_{x,k}^{T} + \boldsymbol{H}_{c,k}\boldsymbol{P}_{cc,k}^{-}\boldsymbol{H}_{c,k}^{T} + \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}_{v,k}^{T} \\ & + \boldsymbol{H}_{x,k}\boldsymbol{P}_{xc,k}^{-}\boldsymbol{H}_{c,k}^{T} + \boldsymbol{H}_{c,k}(\boldsymbol{P}_{xc,k}^{-})^{T}\boldsymbol{H}_{x,k}^{T}\,.\end{aligned} \tag{3.48f}$$

Proof is given in Appendix C.7.

Recall that no measurement update is performed on $\boldsymbol{m}_{c,k}^{-}$ or $\boldsymbol{P}_{cc,k}^{-}$, such that

$$\begin{aligned}\boldsymbol{m}_{c,k}^{+} &= \boldsymbol{m}_{c,k}^{-} \\ \boldsymbol{P}_{cc,k}^{+} &= \boldsymbol{P}_{cc,k}^{-}\,.\end{aligned}$$

That is, as consider parameters, the incoming data are not utilized to estimate the consider parameters; the data are only used to estimate the states. This completes the Bayesian consider measurement update and serves as the Bayesian analog to the MMSE measurement update derived in Section 3.1.

Remark 3.9 (The Bayesian Analog of MMSE Consider Filters)**.** *Inspection of these results permits the conclusion that these expressions are, in fact, equivalent to those derived under MMSE principles in Sections 3.1.2 and 3.1.3. It may not be immediately apparent, but to demonstrate this, the results of Sections 3.1.2 and 3.1.3 are manipulated to produce the claimed results.*

Claim of Eq. (3.48a): *Starting with Eq. (3.5a), write*

$$\begin{aligned}\boldsymbol{m}_{x,k}^{+} &= \boldsymbol{m}_{x,k}^{-} + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{m}_{z,k}^{-}) \\ &= \boldsymbol{m}_{x,k}^{-} + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{m}_{x,k}^{-}, \boldsymbol{m}_{c,k}^{-}, \boldsymbol{0}_{n_v\times 1})) && \text{(substituting Eq. (3.13a))} \\ &= \boldsymbol{m}_{x,k}^{-} + \boldsymbol{K}_{x,k}(\boldsymbol{z}_k - \boldsymbol{H}_{x,k}\boldsymbol{m}_{x,k}^{-} - \boldsymbol{H}_{c,k}\boldsymbol{m}_{c,k}^{-}) && \text{(modeling from Sec. 3.3.1.2)}\end{aligned}$$

as claimed.

Claim of Eq. (3.48b): *Starting with Eq. (3.9c), write*

$$\begin{aligned}
\boldsymbol{P}^{+}_{xx,k} &= \boldsymbol{P}^{-}_{xx,k} - \boldsymbol{K}_{x,k}\boldsymbol{P}^{-}_{zz,k}\boldsymbol{K}^{T}_{x,k} \\
&= \boldsymbol{P}^{-}_{xx,k} - \boldsymbol{P}^{-}_{xz,k}(\boldsymbol{P}^{-}_{zz,k})^{-1}(\boldsymbol{P}^{-}_{xz,k})^{T} && \text{(using Eq. (3.10))} \\
&= \boldsymbol{P}^{-}_{xx,k} - \boldsymbol{K}_{x,k}(\boldsymbol{P}^{-}_{xx,k}\boldsymbol{H}^{T}_{x,k} + \boldsymbol{P}^{-}_{xc,k}\boldsymbol{H}^{T}_{c,k})^{T} && \text{(using Eqs. (3.10) and (3.13c))} \\
&= (\boldsymbol{I}_{n_x\times n_x} - \boldsymbol{K}_{x,k}\boldsymbol{H}_{x,k})\boldsymbol{P}^{-}_{xx,k} - \boldsymbol{K}_{x,k}\boldsymbol{H}_{c,k}(\boldsymbol{P}^{-}_{xc,k})^{T}
\end{aligned}$$

as claimed.

Claim of Eq. (3.48c): *Starting with Eq. (3.9e), write*

$$\begin{aligned}
\boldsymbol{P}^{+}_{xc,k} &= \boldsymbol{P}^{-}_{xc,k} - \boldsymbol{K}_{x,k}(\boldsymbol{P}^{-}_{cz,k})^{T} \\
&= \boldsymbol{P}^{-}_{xc,k} - \boldsymbol{K}_{x,k}[(\boldsymbol{P}^{-}_{xc,k})^{T}\boldsymbol{H}^{T}_{x,k} + \boldsymbol{P}^{-}_{cc,k}\boldsymbol{H}^{T}_{c,k}]^{T} && \text{(using Eq. (3.13d))} \\
&= (\boldsymbol{I}_{n_x\times n_x} - \boldsymbol{K}_{x,k}\boldsymbol{H}_{x,k})\boldsymbol{P}^{-}_{xc,k} - \boldsymbol{K}_{x,k}\boldsymbol{H}_{c,k}\boldsymbol{P}^{-}_{cc,k}
\end{aligned}$$

as claimed.

The claims of Eqs. (3.48d)–(3.48f) are trivially satisfied.

Therefore, the claim that these Bayesian consider corrector equations are algorithmic equivalents to the MMSE-based consider filtering equations, under the selected assumptions, is supported.

Summary. In summary, Eqs. (3.47) define the mean, covariance, and correlations matrix prediction and Eqs. (3.48) allow the mean, covariance, and correlations matrix measurement update of the Bayesian consider filter. Note that all of the terms necessary to perform subsequent predictor/corrector cycles have been obtained (e.g. a prediction to time t_{k+1} from time t_k to incorporate some new measurement $\boldsymbol{z}_{k+1}$), so this forms the desired recursion. Also note that, following the measurement update, only the conditional mean and covariance of the state are obtained, not the consider parameters. This is true to the spirit of consider analysis, as only the effects of the consider parameters, and their associated uncertainties, on the state estimate and its error covariance are of interest.

3.3.3. Non-Gaussian State Densities. Ho's Bayesian interpretation of the Kalman filter [67] enabled Sorenson to develop the Gaussian sum filter, wherein state (and noise) densities are described by GMs [62]. As the previous section illustrated the use of Lemmas 3.3 and 3.4 to obtain the consider filter using Bayes' rule, the current section describes a similar result to Sorenson but under the premise of consider filtering. By approximating the state density as a GM, systems described by non-Gaussian state densities can be accurately estimated. The following discussion will continue to assume that all noises and consider parameters follow Gaussian distributions. This is not a requirement to obtain a closed recursion, but it is presented this way for ease of exposition. Accommodating GM noises and GM consider parameters is simply a matter of added notational, and computational, complexity.

3.3.3.1. Predictor. Consider the dynamical system described in Section 3.3.1.2. Assuming that the joint posterior distribution at time t_{k-1} is an L-component GM of the form[9]

$$p(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}|\boldsymbol{Z}_{1:k-1}) = \sum_{\ell=1}^{L} w^{+}_{\ell,k-1} p_g \left(\begin{bmatrix} \boldsymbol{x}_{k-1} \\ \boldsymbol{c}_{k-1} \end{bmatrix} ; \begin{bmatrix} \boldsymbol{m}^{+}_{x,\ell,k-1} \\ \boldsymbol{m}^{+}_{c,k-1} \end{bmatrix}, \begin{bmatrix} \boldsymbol{P}^{+}_{xx,\ell,k-1} & \boldsymbol{P}^{+}_{xc,\ell,k-1} \\ (\boldsymbol{P}^{+}_{xc,\ell,k-1})^T & \boldsymbol{P}^{+}_{cc,k-1} \end{bmatrix} \right) \tag{3.49}$$

and that the consider parameter vector is Gaussian distributed with mean $\boldsymbol{m}^{+}_{c,k-1}$ and covariance $\boldsymbol{P}^{+}_{cc,k-1}$, an application of Lemma 3.3 to the Chapman-Kolmogorov equation (Eq. (3.38)) yields the joint prior distribution at time t_k as

$$p(\boldsymbol{x}_k, \boldsymbol{c}_k|\boldsymbol{Z}_{1:k-1}) = \sum_{\ell=1}^{L} w^{-}_{\ell,k} p_g \left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{c}_k \end{bmatrix} ; \begin{bmatrix} \boldsymbol{m}^{-}_{x,\ell,k} \\ \boldsymbol{m}^{-}_{c,k} \end{bmatrix}, \begin{bmatrix} \boldsymbol{P}^{-}_{xx,\ell,k} & \boldsymbol{P}^{-}_{xc,\ell,k} \\ (\boldsymbol{P}^{-}_{xc,\ell,k})^T & \boldsymbol{P}^{-}_{cc,k} \end{bmatrix} \right), \tag{3.50}$$

where

$$w^{-}_{\ell,k} = w^{+}_{\ell,k-1} \tag{3.51a}$$

$$\boldsymbol{m}^{-}_{x,\ell,k} = \boldsymbol{F}_{x,k-1} \boldsymbol{m}^{+}_{x,\ell,k-1} + \boldsymbol{F}_{c,k-1} \boldsymbol{m}^{+}_{c,k-1} \tag{3.51b}$$

[9]Here, the subscript ℓ is an indexing variable, such that, for example, $\boldsymbol{m}^{+}_{x,\ell,k-1}$ is the ℓ^{th} component mean in the GM density. Note that, due to (relaxable) assumptions of Gaussianity, $\boldsymbol{m}^{+}_{c,k-1}$ and $\boldsymbol{P}^{+}_{cc,k-1}$ have no such indexing variable; that is, they are identical for each component.

$$\boldsymbol{m}^{-}_{c,k} = \boldsymbol{G}_{c,k-1}\boldsymbol{m}^{+}_{c,k-1} \tag{3.51c}$$

$$\begin{aligned}\boldsymbol{P}^{-}_{xx,\ell,k} = {} & \boldsymbol{F}_{x,k-1}\boldsymbol{P}^{+}_{xx,\ell,k-1}\boldsymbol{F}^{T}_{x,k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{P}^{+}_{cc,k-1}\boldsymbol{F}^{T}_{c,k-1} + \boldsymbol{F}_{w,k-1}\boldsymbol{P}_{ww,k-1}\boldsymbol{F}^{T}_{w,k-1} \\ & + \boldsymbol{F}_{x,k-1}\boldsymbol{P}^{+}_{xc,\ell,k-1}\boldsymbol{F}^{T}_{c,k-1} + \boldsymbol{F}_{c,k-1}(\boldsymbol{P}^{+}_{xc,\ell,k-1})^{T}\boldsymbol{F}^{T}_{x,k-1}\end{aligned} \tag{3.51d}$$

$$\boldsymbol{P}^{-}_{cc,k} = \boldsymbol{G}_{c,k-1}\boldsymbol{P}^{+}_{cc,k-1}\boldsymbol{G}^{T}_{c,k-1} + \boldsymbol{G}_{u,k-1}\boldsymbol{P}_{uu,k-1}\boldsymbol{G}^{T}_{u,k-1} \tag{3.51e}$$

$$\boldsymbol{P}^{-}_{xc,\ell,k} = \boldsymbol{F}_{x,k-1}\boldsymbol{P}^{+}_{xc,\ell,k-1}\boldsymbol{G}^{T}_{c,k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{P}^{+}_{cc,k-1}\boldsymbol{G}^{T}_{c,k-1}\,. \tag{3.51f}$$

Proof. This result follows in much the same way as the Gaussian case in Section 3.3.2.1 but with Eq. (3.49) as the posterior density at t_{k-1}. ■

The prior density of only $\boldsymbol{x}_k$, i.e. without dependence on the parameters, can be obtained by factoring each GM component as shown at the end of Section 3.3.2.1 and integrating over $\boldsymbol{c}_k$.

As is common in GM-based methods, each mixand's weight is kept constant over the propagation from t_{k-1} to t_k. This is only absolutely true for systems where the underlying dynamics are linear; however, it is an approximation that does not wind up being hugely damaging to the performance of many Gaussian sum filter applications [53, 62, 68, 74, 75, 76]. Research has been presented, such as in [77], to adapt the component weights to changes in the probability density function due to system nonlinearities, and such approaches can be applied here if the developer so chooses.

3.3.3.2. Corrector. A measurement $\boldsymbol{z}_k$ according to likelihood Eq. (3.44) is received. It is desired to improve the statistical description of the state given by the prior distribution, a GM obtained via the prediction step, via this measurement to acquire the posterior state GM distribution. Since estimates for the consider parameters are not of interest, the marginal density for the state is obtained via a marginalization of Bayes' rule, described in Eq. (3.41).

Taking the prior joint density to be given by Eq. (3.50), use of Eq. (3.41) and Lemmas 3.3 and 3.4 yields the marginal posterior state density as the GM

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k}) = \sum_{\ell=1}^{L} w^{+}_{\ell,k}\, p_g(\boldsymbol{x}_k\,;\, \boldsymbol{m}^{+}_{x,\ell,k}, \boldsymbol{P}^{+}_{xx,\ell,k})\,, \tag{3.52}$$

where

$$w_{\ell,k}^{+} = \frac{w_{\ell,k}^{-} q_{\ell,k}(\boldsymbol{z}_k)}{\sum_{j=1}^{L} w_{j,k}^{-} q_{j,k}(\boldsymbol{z}_k)} \tag{3.53a}$$

$$q_{\ell,k}(\boldsymbol{z}_k) = p_g(\boldsymbol{z}_k \,;\, \boldsymbol{H}_{x,k}\boldsymbol{m}_{x,\ell,k}^{-} + \boldsymbol{H}_{c,k}\boldsymbol{m}_{c,k}^{-}, \boldsymbol{P}_{zz,\ell,k}) \tag{3.53b}$$

$$\boldsymbol{m}_{x,\ell,k}^{+} = \boldsymbol{m}_{x,\ell,k}^{-} + \boldsymbol{K}_{x,\ell,k}(\boldsymbol{z}_k - \boldsymbol{H}_{x,k}\boldsymbol{m}_{x,\ell,k}^{-} - \boldsymbol{H}_{c,k}\boldsymbol{m}_{c,k}^{-}) \tag{3.53c}$$

$$\boldsymbol{P}_{xx,\ell,k}^{+} = (\boldsymbol{I}_{n_x \times n_x} - \boldsymbol{K}_{x,\ell,k}\boldsymbol{H}_{x,k})\boldsymbol{P}_{xx,\ell,k}^{-} - \boldsymbol{K}_{x,\ell,k}\boldsymbol{H}_{c,k}(\boldsymbol{P}_{xc,\ell,k}^{-})^T \tag{3.53d}$$

$$\boldsymbol{P}_{xc,\ell,k}^{+} = (\boldsymbol{I}_{n_x \times n_x} - \boldsymbol{K}_{x,\ell,k}\boldsymbol{H}_{x,k})\boldsymbol{P}_{xc,\ell,k}^{-} - \boldsymbol{K}_{x,\ell,k}\boldsymbol{H}_{c,k}\boldsymbol{P}_{cc,k}^{-} \tag{3.53e}$$

$$\boldsymbol{K}_{x,\ell,k} = \boldsymbol{P}_{xz,\ell,k}\boldsymbol{P}_{zz,\ell,k}^{-1} \tag{3.53f}$$

$$\boldsymbol{P}_{xz,\ell,k}^{-} = \boldsymbol{P}_{xx,\ell,k}^{-}\boldsymbol{H}_{x,k}^{T} + \boldsymbol{P}_{xc,\ell,k}^{-}\boldsymbol{H}_{c,k}^{T} \tag{3.53g}$$

$$\begin{aligned}\boldsymbol{P}_{zz,\ell,k}^{-} = {} & \boldsymbol{H}_{x,k}\boldsymbol{P}_{xx,\ell,k}^{-}\boldsymbol{H}_{x,k}^{T} + \boldsymbol{H}_{c,k}\boldsymbol{P}_{cc,k}^{-}\boldsymbol{H}_{c,k}^{T} + \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}_{v,k}^{T} \\ & + \boldsymbol{H}_{x,k}\boldsymbol{P}_{xc,\ell,k}^{-}\boldsymbol{H}_{c,k}^{T} + \boldsymbol{H}_{c,k}(\boldsymbol{P}_{xc,\ell,k}^{-})^T\boldsymbol{H}_{x,k}^{T}\,.\end{aligned} \tag{3.53h}$$

Proof. By following the same procedure as with the Gaussian consider corrector of Section 3.3.2.2, instead using the GM in Eq. (3.50) as the prior density, one arrives at nearly the same result using Lemma 3.4, but that the term $w_{\ell,k}^{-} q_{\ell,k}(\boldsymbol{z}_k)$ appears, where $q_{\ell,k}(\boldsymbol{z}_k)$ is as given in Eq. (3.53b). In addition, an application of Lemma 3.4 in the normalization constant term produces

$$\eta_k = \sum_{j=1}^{L} w_{j,k}^{-} q_{j,k}(\boldsymbol{z}_k)\,.$$

Therefore, one can conclude that

$$w_{\ell,k}^{+} = \frac{w_{\ell,k}^{-} q_{\ell,k}(\boldsymbol{z}_k)}{\sum_{j=1}^{L} w_{j,k}^{-} q_{j,k}(\boldsymbol{z}_k)}\,,$$

and the rest of Eqs. (3.53) follow in the same way as the proof in Section 3.3.2.2. ■

Similar to the Gaussian case, no update is performed on $\boldsymbol{m}_{c,k}^{-}$ or $\boldsymbol{P}_{cc,k}^{-}$, such that

$$\boldsymbol{m}_{c,k}^{+} = \boldsymbol{m}_{c,k}^{-}$$
$$\boldsymbol{P}_{cc,k}^{+} = \boldsymbol{P}_{cc,k}^{-}\,.$$

Summary. In summary, Eqs. (3.50) and (3.51) define each component's weight, mean, covariance, and correlations matrix prediction step, and Eqs. (3.52) and (3.53) allow each component's weight, mean, covariance, and correlations matrix measurement update. Note that the presented equations are the previously developed equations for Gaussian state densities applied to each GM component but with an additional update step for the GM weights. Given that the prescribed assumptions are satisfied, the recursion is closed and results in a GM formulation of the consider filter. Since a GM distribution is produced, the recursion is closed and can be used to recursively update the target state density.

3.3.4. Numerical Example. Consider again the ballistic trajectory example presented in Section 3.2.6 on pp. 99. Now, however, the initial statistics of the vehicle are taken to be highly non-Gaussian as seen in Figure 3.10. These types of distributions arise when very accurate range and angle information is available, so here this serves to simulate the tracking radar using measurement data to initialize the tracking filter.[10] In Figure 3.10, the conditional mean is denoted as a circle, and 1σ, 2σ, and 3σ contours for a Gaussian distribution parameterized according to the conditional mean and covariance are plotted as dotted lines. These statistics are clearly non-Gaussian, but they illustrate how the mean and covariance, in this case, do not well-describe the initial non-Gaussian density.

100 Monte Carlo trials are performed to assess the differences between filters that would rely solely on mean and covariance (or, in this case, mean and square-root factor) and filters that utilize a non-Gaussian parameterization such as a GM. To that end, the linearization-based square-root consider filter of Section 3.1.3 (called the C-SREKF in the example of Section 3.2.6) is compared to a square-root consider formulation of the GM

[10]This lensing effect, the "banana" shape, occurs when Gaussian distributions in a polar space (such as the radar's range and angle data) are transformed into Cartesian coordinates. The lens traces a path of approximately equal range values (a circle traced around the observer) and angle information indicates a location on that circle that is more likely.

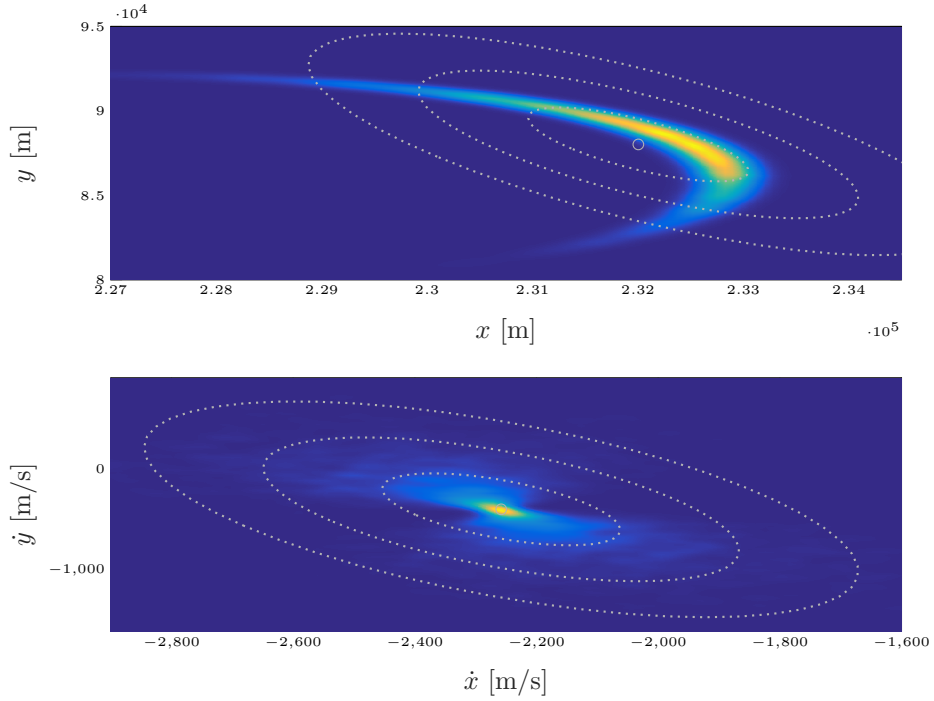
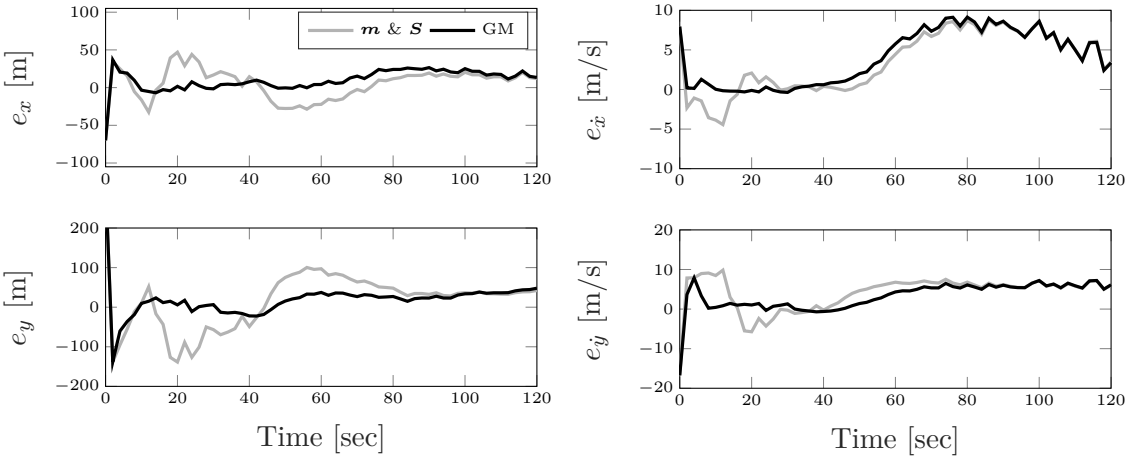


Figure 3.10. Initial non-Gaussian density and conditional mean and covariance visualization for the ballistic trajectory.

consider filter from the previous section. The filter that relies on mean and square-root factor alone, referred to in later figures as "$\boldsymbol{m}$ & $\boldsymbol{S}$" is initialized according to the conditional mean and square-root factor of the initial GM in Figure 3.10, and the GM consider filter is initialized with the GM itself. Otherwise, all system parameters are identical to the example of Section 3.2.6. In the GM filter, the state estimate for computing error at each time step is taken as the conditional mean of the posterior GM density (computed via the method of moments). The method of moments for GMs is briefly outlined in Appendix B.

As before, 100 Monte Carlo trials are performed with all noises, parameter values, and trajectories resampled on each trial, and the error statistics are collected. The mean errors for position and velocity are presented in Figures 3.11a and 3.11b, respectively, and these figures indicate that both filters are, on average, able to converge upon approximately zero estimation errors. The GM filter produces somewhat better (i.e. closer to zero) estimation errors, but both filters begin to perform identically as more and more data is processed. This is an expected result, as the central limit theorem states that repeated

data addition/processing will ultimately produce Gaussian statistics, and Gaussian densities are fully parameterized by mean and covariance (or, in this case, square-root factor)! Therefore, eventually, both filters should be expected to perform similarly, and this is seen in this case.

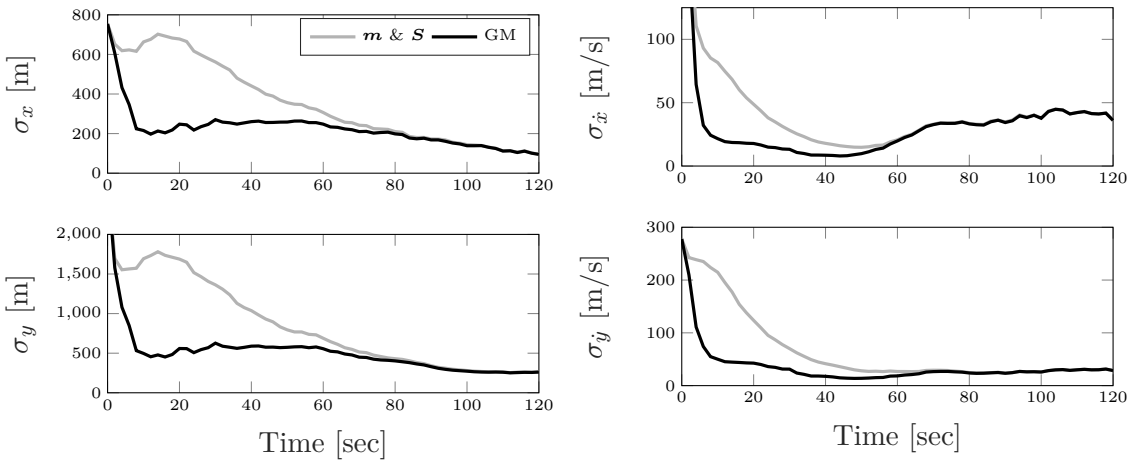


(a) Position. (b) Velocity.

Figure 3.11. Monte Carlo errors for the ballistic target.

On the other hand, the Monte Carlo sample error standard deviations shown in Figures 3.12a and 3.12b indicate drastic differences in the transient performance of the two filters. The GM filter produces substantially lower error uncertainties than the filter that relies solely upon mean and square-root factor for about half the duration of the simulation (before the two filters converge upon each other). So, both filters, while quite different, arrive at approximately the same result once the effect of the central limit theorem takes hold (i.e. approximate Gaussianity is obtained), but the GM filter produces estimates with considerably less *cumulative* error and associated uncertainty.

It is common to mistake a filter that relies upon mean and covariance (or square-root factor) as being one that is restricted to problems where the underlying state or noise statistics are Gaussian. One may be inclined to draw this conclusion from looking at the Gaussian density results in Section 3.3.2, but recall that the very same expressions were produced even earlier in this section using MMSE principles without once invoking the word "Gaussian." In this case, the same result can be obtained using two very different derivations. This is emphasized here because stating that a filter estimating mean and



(a) Position. (b) Velocity.

Figure 3.12. Monte Carlo standard deviations for the ballistic target.

square-root factor approximates the non-Gaussian density in Figure 3.10 as Gaussian would, perhaps, be a limiting perspective. Rather, it estimates the first two statistical moments of the non-Gaussian density directly (be it Gaussian, uniform, gamma, beta, exponential, or what have you), whereas the GM filter aims to estimate a parameterization of that density itself. In this sense, the mean and covariance/square-root filters are quite general, even though they are commonly, and incorrectly, described as requiring Gaussian statistics. They make no requirement that anything be Gaussian, and thus are viable candidates for use in non-Gaussian problems. Even though they may be outperformed by the more complex non-Gaussian filters, it is important to be aware of every tool in the "toolbox" for different classes of problems, particularly in cases where computational resources are limited.

4. MULTITARGET FILTERING FOR NAVIGATION

In all of the discussion leading to this point, the goal has been to estimate a vector-valued state as it evolves through time and is observed by a similarly vector-valued observation process. Many problems, however, involve estimating the states of multiple targets simultaneously, where each of the targets evolves within a dynamic environment that contains uncertain model parameters and is observed through a measurement process that also contains uncertain model parameters.[1] The tracking of multiple targets in a single framework has been a problem that has generated a great deal of research interest for some time and inherently relies on accurate knowledge of the uncertainties introduced by parameter errors to maintain accurate state estimates of the tracked targets. Many of the classical methods for multitarget tracking rely on heuristics to extend single-target tracking techniques, such as the MMSE filter of Section 2, directly into the multitarget domain [74, 78, 79]. These heuristics effectively gate the incoming observations by some reference, such as nearest-neighbor association or Mahalanobis distance based on assumed measurement statistics, in order to associate the measurements to some *a priori* knowledge of the targets, such as is the case with joint probabilistic data association [79] and traditional techniques for multiple hypothesis tracking [80]. Ideally, a true multitarget framework is desired, as opposed to gating the mechanisms of a single-target tracking algorithm.

A new class of multitarget filtering techniques has emerged that leverages recent developments in random finite sets (RFSs) and finite set statistics (FISST) and casts off the heuristics-based approaches of the past in place of a rigorous, fully probabilistic and model-based estimation framework [81, 82, 83]. In this environment, the multitarget state, containing the states of all the tracked objects, and the multitarget observation, containing all of the observations of these objects, are modeled as RFSs. In essence, an RFS is a collection of vectors whose spatial characteristics are all unknown, and the total number

[1]Note that moving forward, the terminology "single target" will often be used to describe those filters that estimate a single state, $\boldsymbol{x}_k$. In the context of, say, navigation problems, the terminology "target" seems strange, but this language is inspired by convention and is useful to contrast these methods with multitarget filters.

of random vectors within that set is also unknown (and, therefore, both target states and number of targets must be estimated simultaneously). Leveraging this construction, in conjunction with FISST, has allowed the development of a new multitarget filter: the multitarget Bayes filter. Serving as a multitarget analog to the recursion formed by the Chapman-Kolmogorov equation and Bayes' rule, the multitarget Bayes filter processes the collected multitarget observations to refine an estimate of the multitarget state. In practice, implementing such a filter is an inherently challenging task, as multitarget estimation has a number of particularly frustrating problems: number of targets is, in general, unknown and must be estimated, target states must be estimated under uncertain data association events, detection probabilities and false returns must be modeled, target survival and death must be modeled, spawning events must be considered, the addition of new targets into a surveillance scene must be modeled, etc. Indeed, the multitarget Bayes filter, thanks to the immense complexity thrust upon it by a very general and challenging modeling environment, is intractable in general and thus requires approximation. The first attempts to approximate this recursion were brute-force sample-based implementations that, while they can be made to work well under certain assumptions such as in [84], make practical implementation largely infeasible.

Enticed by the potential of multitarget Bayesian filtering, and stymied by its general intractability, researchers began to contemplate more refined approximations to these techniques. Stein and Winter, of Los Alamos National Laboratories, and Tenner, of Alphatech Corporation, began to theorize a desirable class of nonnegative functions, that they called a probability hypothesis surface, with a single property: that the integral of that surface over some region in space equals the number of targets within that region [85, 86]. In his portion of Goodman's textbook [82], R. P. S. Mahler developed a rigorous interpretation of this probability hypothesis surface, instead dubbing it the probability hypothesis density (PHD), and explored a number of its interesting features, implications, and connection with

what would later come to be commonly called FISST. Mahler would go on to truly champion the PHD approach and presented a thorough discussion on a proposed PHD filter in the highly referenced work of [87].[2]

The main principle of the PHD filter is inspired by, and not unlike, the MMSE principles presented in Section 2 and the first half of Section 3, despite being firmly rooted to a Bayesian interpretation. While the resulting filter was groundbreaking, the concept is simple: instead of estimating the complex density produced by Bayes' rule, directly estimate that density's statistical moments. The PHD is the first-order statistical moment of such densities and serves as the multitarget analog to the vector-valued mean. In fact, the PHD filter is analogous to an MMSE filter that only estimates the mean, also known as the fixed-gain Kalman filter. Unlike the standard mean, however, the PHD is a "peaky" surface defined on the target state space, where peaks correspond to probable target locations in that state space. The PHD filter would later be generalized to contain higher-order information by estimating the full cardinality (i.e. number of targets) distribution jointly with the PHD, resulting in the cardinalized PHD (CPHD) filter [88]. While incurring a significant increase in theoretical and computational complexity, the CPHD filter solves the primary failing of the PHD filter, and that is in its poor cardinality estimation. Very recently, Schlangen et al. derived a second-order PHD filter that addresses the poor cardinality estimation of the PHD filter at a substantially lower cost than the CPHD filter [89].[3]

The PHD and CPHD filters have become popular for many applications due to their tractability, and their GM approximations have received particular attention [69, 70]. While this first-order moment approximation comes with an inherent loss of information, it provides a great deal of feasibility, and its use has been demonstrated in many areas [90, 91, 92, 93, 94, 95, 96, 97, 98]. This is far from anything that could be called a complete survey of applications, but the intended point should be clear: the applications of these filters are numerous and their results promising. They are not without their shortcomings and criticisms, however. Despite offering immense practical value, PHD-based filters

[2]Due to its use within and relationship to point process literature, the terminology "intensity" or "intensity function" is often used to describe the PHD. This terminology will be used interchangeably in this dissertation.

[3]Interestingly, this second-order PHD filter, in a way, completes the analogy between the PHD filter and the MMSE filter that estimates mean and covariance.

are critically hampered in that their state estimation requires mode-finding of the PHD itself, a procedure that is known to be expensive and unpredictable. The most commonly utilized technique to avoid explicit mode finding in GM implementations is to select mixture components with weights over some prescribed threshold as estimates, but this tends to suffer in realistic scenarios (such as closely spaced targets).

Other researchers sought to improve multitarget tracking capabilities by, rather than estimating statistical moments, estimating parameters of the densities produced by multitarget Bayes filter. This approach is analogous to the Bayesian approach to Kalman filtering as described previously in Section 3.3. This necessitated concocting new functions to characterize the multitarget distributions, and among the earliest successes in this arena was that of the multitarget, multi-Bernoulli filter (as is thoroughly explored in B.T. Vo's dissertation [71]), but it demonstrated varying degrees of success and, ultimately, years passed before a new paradigm was developed to tackle this problem. Vo and Vo devised of a new class of RFSs, called labeled RFSs, that, when carefully formulated, produces a closed-form solution to the multitarget Bayes filter under fairly standard modeling assumptions. This filter, called the δ-generalized labeled multi-Bernoulli (δ-GLMB) filter [72, 73], has generated vast amounts of interest by the estimation community. Substantial effort has been applied to producing very efficient implementations of the δ-GLMB filter, including the principled approximation proposed by Reuter [99] or the scalable formulation of Beard [100]. Again, no survey is attempted here, but a curious reader can refer to Mahler's textbooks, [83] and [101], for a thorough survey of available methods and their applications.

Despite this vast amount of research, consider filtering (as discussed at length in Section 3) has yet to be formally generalized to the multitarget domain. Inspection of the predictor/corrector relationships for GM implementations of multitarget filters indicates that they require the very same types of calculations as traditional filters (for example, the update gain is identically constructed as the product of a cross covariance and a matrix inverse) and are similarly subject to the effects of numerical imprecision. If these filters are to be reliably applied, as required by real-time tracking (such as problems like the recurring ballistic trajectory example) and onboard navigation (as will be discussed in Section 4.3 and Section 5), numerical stability must be a significant focus when designing a filter. This makes

numerical hardening techniques such as consider filtering very attractive for multitarget tracking, and thus consider filtering is generalized to multitarget consider filtering in the following discussion.

Section Structure. This discussion opens in Section 4.1 by delivering brief preliminaries on FISST and multitarget inference with Bayes' rule. These points prove critical to a complete discussion of the topics that follow.

Section 4.2 goes on to extend the principles of consider filtering to the multitarget domain. Section 3.3 demonstrated the Bayesian analog to MMSE consider filtering, enabling a new class of single-target consider filters for Gaussian and non-Gaussian state densities, and the following developments aim to further leverage those results by generalizing them to multitarget filters. In particular, a new PHD filter, the consider PHD filter, is derived using GM approximation. Rather than subsequently (and redundantly) deriving consider forms of other multitarget filters, like the CPHD filter, the δ-GLMB filter, etc., sufficient details are provided such that the various forms of these filters can be obtained using this dissertation..

Section 4.3 then posits how to use these new multitarget consider filters for navigation applications. Inspired by the work in FISST-based simultaneous localization and mapping (SLAM) of Mullane [102], approximate methods for terrain aiding with multitarget filters are developed and thoroughly explored. Due to consider filtering's widespread use within navigation, a terrain aiding application helps to further motivate the generalization of consider filtering to the multitarget domain. The terminology "terrain aiding" is used to imply that a vehicle collects some observations of its environment, such as rocks, walls, other vehicles, etc., and uses these observations to meaningfully improve a navigation solution.[4] Under this interpretation, it becomes clear why multitarget tracking is required: given a collection of environment-sourced data, such as an image, a lidar scan, etc., and no other information, how does one process the data? Certainly, some *a priori* information and computer-based processing can be applied, such as the way star cameras utilize template matching to match pixels to known stars, but what if the background is unknown? What if

[4]It is conceded that another vehicle is, perhaps, not best classified as "terrain", but the result is the same: observations of a vehicle's environment are collected and used for navigation. This terminology has heritage within ballistic missile and spaceflight research and is therefore adopted.

the observation process cannot be modeled with traditional filtering techniques? What if, for example, a lunar descent vehicle passes over an uncertain surface and collects images?[5] Must one rely entirely upon an *a priori* reference map, or is another approach possible to meaningfully update a navigation solution?

The section continues in Section 4.4 by proposing the use of decentralized fusion to augment, rather than replace, single-target filtering architectures of the type presented in Sections 2 and 3. That is, if a sensor suite contains sensors that provide terrain data (i.e. multitarget data, such as terrain cameras) as well as traditional sensors (i.e. single-target data, such as GPS or an altimeter), how can one process these data separately and marry the two? Furthermore, why would one want to separate the two architectures rather than process everything in a single framework? These questions are explored, and a practical fusion technique is presented.

The concludes in Section 4.5 by deriving a new PHD filter that, rather than estimating one set, simultaneously estimates multiple sets, each with their own models and properties. This has advantages in cases where the targets have different dynamics (such as a surveillance region containing both static and dynamic targets) or different observational qualities (such as reflective and non-reflective targets). Furthermore, what if, for the purposes of a given mission design, one is only concerned about a subset of the total collection of targets? The derived filter is throughly tested in a number of simulations to determine its salient elements, and the new filter is applied to a navigation problem to evaluate its performance.

Remark 4.1. *As a general note, a reader will notice that the following describes full covariance formulations of the derived filters rather than the extolled square-root forms of earlier sections. This is because this reasonably simplifies notation and the author feels that the path to a square-root formulation of any of these filters has been adequately mapped in both Sections 2.3 and 3.2.*

[5] This case is explored as one candidate application in Section 5.

4.1. PRELIMINARIES ON FINITE SET STATISTICS

Suppose that at some instant k, there are N_k vectors $\boldsymbol{x}_{k,i} \in \mathbb{R}^{n_x}$ describing dynamically evolving targets, and define an RFS $\boldsymbol{X}_k$, called the multitarget state, such that

$$\boldsymbol{X}_k = \{\boldsymbol{x}_{k,1}, \ldots, \boldsymbol{x}_{k,N_k}\}.$$

Note that $\boldsymbol{X}_k$ contains a (potentially) time-varying number of targets, such that targets may disappear, multiply, or appear at each time instance. Further suppose that $\boldsymbol{Z}_k$ is an RFS containing vector-valued observations, such that

$$\boldsymbol{Z}_k = \{\boldsymbol{z}_{k,1}, \ldots, \boldsymbol{z}_{k,D_k}\}$$

reported to have been generated by $\boldsymbol{X}_k$. The RFS $\boldsymbol{Z}_k$ potentially contains not only observations generated by $\boldsymbol{X}_k$ but also may contain spurious returns, herein referred to as clutter. The process that generates $\boldsymbol{Z}_k$ is faulty in that detections are not guaranteed (i.e. missed detections are possible even if some $\boldsymbol{x} \in \boldsymbol{X}_k$ should be expected to generate a sensor return) and that each $\boldsymbol{z} \in \boldsymbol{Z}_k$ is subject to corruption by random noise.

It is ultimately sought to estimate the states $\boldsymbol{x} \in \boldsymbol{X}_k$ and the cardinality of $\boldsymbol{X}_k$, denoted as $|\boldsymbol{X}_k|$, as $\boldsymbol{X}_k$ evolves through time and measurements $\boldsymbol{Z}_k$ are collected. This is a joint estimation problem, joint in that both the target states and set cardinality must be estimated simultaneously. On a theoretical level, this task is solved entirely with the multitarget Bayes filter, which is a recursion formed by the multitarget analogs of the Chapman-Kolmogorov equation and Bayes' rule. Given some *a posteriori* multitarget density at $k-1$, denoted as $\pi(\boldsymbol{X}_{k-1}|\boldsymbol{Z}_{1:k-1})$, the *a priori* multitarget density is obtained via the forecasting step

$$\pi(\boldsymbol{X}_k|\boldsymbol{Z}_{1:k-1}) = \int f(\boldsymbol{X}_k|\boldsymbol{X}_{k-1})\pi(\boldsymbol{X}_{k-1}|\boldsymbol{Z}_{1:k-1})\delta\boldsymbol{X}_{k-1}\,,$$

where $f(\boldsymbol{X}_k|\boldsymbol{X}_{k-1})$ is an appropriate multitarget transition density and $\int f(\boldsymbol{Y})\delta\boldsymbol{Y}$ denotes the set integral given by

$$\int f(\boldsymbol{Y})\delta\boldsymbol{Y} = f(\emptyset) + \sum_{n=1}^{\infty} \frac{1}{n!} \int f(\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}) \mathrm{d}\boldsymbol{y}_1 \cdots \mathrm{d}\boldsymbol{y}_n \,.$$

Effectively, each term in the set integral's expression accounts for a candidate cardinality of a set, everywhere from the set being empty, i.e. $\emptyset$, to having a countably infinite number of elements. With prediction complete, $\boldsymbol{Z}_k$ is processed to meaningfully improve the *a priori* understanding of $\boldsymbol{X}_k$. This is accomplished using the multitarget analog to Bayes' rule, given by

$$\pi(\boldsymbol{X}_k|\boldsymbol{Z}_{1:k}) = \frac{g(\boldsymbol{Z}_k|\boldsymbol{X}_k)\pi(\boldsymbol{X}_k|\boldsymbol{Z}_{1:k-1})}{\int g(\boldsymbol{Z}_k|\boldsymbol{W}_k)\pi(\boldsymbol{W}_k|\boldsymbol{Z}_{1:k-1})\delta\boldsymbol{W}_k} \,,$$

where $g(\boldsymbol{Z}_k|\boldsymbol{X}_k)$ is an appropriate multitarget likelihood function to model the process that generated $\boldsymbol{Z}_k$ and the integral is again a set integral.

It doesn't take long to identify that this looks precisely like the ubiquitous Chapman-Kolmogorov/Bayes' rule predictor/corrector recursion for single-target tracking but with set arguments and functions replaced appropriately. As mentioned in the introduction to this section, this serves as the basis for all practical multitarget filtering that utilizes RFS and FISST theory. While this recursion is generally intractable, particularly due to the required set integration, it serves as the starting point for all of the aforementioned filtering techniques, be their derivations intensity-based (i.e. finding multitarget moments, such as with the PHD and CPHD filters) or density-based (i.e. prescribing a form of the multitarget density, such as the δ-GLMB filter).

Many classes of RFS, each consisting of an explicit form for their corresponding multitarget density, have been proposed for use in multitarget filtering. The Poisson RFS is generally the most popular, where targets are treated as being independent and identically distributed (i.i.d.), and the cardinality of the RFS is distributed according to a Poisson distribution [87]. As a Poisson distribution's mean is equal to its variance, Poisson RFS exhibit increasing cardinality variance as the total number of targets increases. This unde-

sired characteristic often motivates the use of a more general RFS known as the i.i.d. cluster RFS, where the i.i.d. property is maintained, and the cardinality distribution can take any form [88]. These are the primary tools of the PHD and CPHD filters, but others utilize multi-Bernoulli RFSs that treat each target as having Bernoulli existence probabilities [71]. Others still define very specific structures to associate target densities to sequences of association histories, such as the δ-GLMB RFS. Regardless of the RFS model employed to produce a useful result, it is the multitarget Bayes filter that serves as the springboard to enabling a predictor/corrector recursion for the estimated multitarget state and is nearly always the starting point for subsequent approximation or manipulation once an RFS is prescribed.

4.2. MULTITARGET CONSIDER FILTERING

In this section, a closed-form GM resursion for the consider PHD filter is derived. Afterwards, some important details regarding evaluating and implementing these filters are discussed, and then some comments are made regarding how to handily obtain consider formulations of other common FISST-based filters, such as the CPHD, LMB, and δ-GLMB filters, using the preceding developments. The new consider PHD filter is then verified and evaluated in a numerical simulation to illustrate the advantages of consider filtering in a multitarget context. Rather interestingly, it is found that, in this case, consider filtering not only provides the numerical stability advantages expected by such a technique, but also that estimation performance is enhanced due to some theoretical limitations of the standard PHD filter.

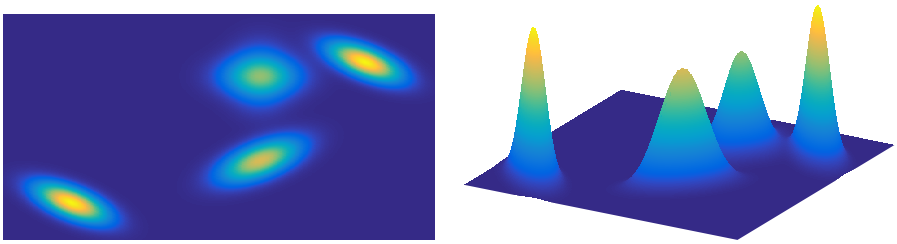
The results of this section are an immediate consequence of the consider filter produced using Bayes' rule in Section 3.3. At first, it may have seemed strange to propose solving the vector-valued (single-target) filtering problem with the intent of later solving the set-valued (multitarget) filtering problem rather than adopting a systematic, "top-down" approach from the outset. However, it just so happens that almost every solution to the multitarget tracking problem can be decomposed to a collection of single-target tracking problems that are cleverly married via the internal mechanisms of the multitarget filter.

Therefore, to solve the multitarget consider filtering problem, one must first solve the single-target consider filtering problem. In fact, the similarities in problem-solving between single-target and multitarget filtering principles are so similar that, in [82], Goodman cites the "Almost Parallel Worlds Principle":

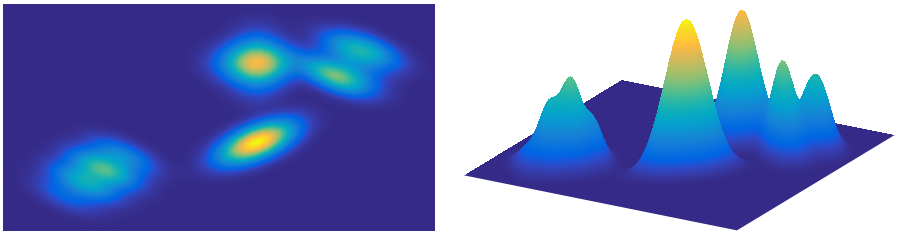
> *"To solve a general multisensor, multitarget data fusion algorithm, first solve the corresponding single-sensor, single-target tracking problem and then directly generalize the mathematics to the multisensor, multitarget case. We say "almost" here because the parallelism is not actually quite complete. Whereas it is possible to add and subtract vectors, there seems to be no analogous addition and subtraction operations for finite sets. This means, among other things, that is not possible to compute expected values of RFSs. Nevertheless, the parallelism is complete enough that, provided one exercises a little care, it is possible to directly generalize a century's worth of knowledge concerning point-variate statistics to multisensor, multitarget data fusion."*

To that end, discussion now turns to leveraging the single-target Bayesian consider filtering strategies developed in previous sections to generalize them to the multitarget domain. In particular, the GM consider PHD filter is produced, and later discussion illustrates how to use these results to obtain various other types of FISST-based multitarget consider filters.

4.2.1. The Consider PHD Filter. Let $v_k^-(\cdot,\cdot)$ and $v_k^+(\cdot,\cdot)$ denote prior and posterior PHDs defined on the joint state space formed between the target state space and the consider parameter state space, respectively. Recall that, when referring to these types of functions, the terms "PHD" and "intensity" are often used interchangeably, and that, while seemingly abstruse to those unfamiliar with the idea, the PHD is a very simple concept: it is simply a positive function with "lumps" within the state space, where lumps correspond to target density, and the entire function integrates over the state space to the number of targets within that space. All other mechanisms are vehicles toward producing a useful filter.



(a) A simple PHD, aerial and perspective views



(b) A more complex PHD, aerial and perspective views

Figure 4.1. Illustration of a pair of representative PHDs for four targets.

For example, consider the pair of PHDs presented in Figure 4.1, where both are PHDs representing four distinct targets. In the top panel, the contribution of each individual target is easy to distinguish, and the peaks of the PHD are easy to identify, yielding logical state estimates. By contrast, the bottom panel illustrates a case where the corresponding state estimates are more ambiguous. For example, there are multiple peaks totaling to at least five, and yet it is known that there are only four targets. Some peaks, such as the central peak, may clearly indicate a state estimate, but the rightmost collection of PHD mass is much more ambiguous. Certainly, one could intuit the tallest peak as a state estimate, but this is a challenging problem to solve in general. Here, it can be seen that the simplicity of the PHD is both a blessing and a curse: it has a perfectly intuitive interpretation, but it may be less-informative than required for some problems. Nevertheless, the PHD has been shown to perform very well in a wide variety of challenging scenarios.

In a similar vein as the development of the GM consider filter from Section 3.3.3, the discussion begins with the joint evolution of the target states and the consider parameters, then applies approximations to obtain a closed recursion, and manipulates the resulting expressions to fall under the consider framework. In the interest of starting with the joint evolution of a target state and the consider parameters, assume that

- each target evolves and generates observations independently of one another,
- the birth RFS, describing spontaneous appearance of new targets, and survival RFS, governing the persistence or disappearing of previously present targets, are independent of each other,
- individual targets are permitted spawn multiple targets,
- clutter is modeled as a Poisson RFS and is generated independently of the tracked objects, and
- the *a priori* multitarget state is a Poisson RFS.

Then, with these assumptions, the PHD filter predictor can be written as

$$v_k^-(\boldsymbol{x}_k, \boldsymbol{c}_k) = v_k^{(S)} + v_k^{(\beta)} + \gamma(\boldsymbol{x}_k, \boldsymbol{c}_k)\,, \tag{4.1}$$

where $v_k^{(S)}$ and $v_k^{(\beta)}$ are elements of the propagated PHD that, respectively, describe surviving and spawned targets and $\gamma(\boldsymbol{x}_k, \boldsymbol{c}_k)$ is the intensity of the RFS describing the objects newly born into the surveillance scene. For compactness, the explicit dependence on the consider parameter vector $\boldsymbol{c}_k$ is shortened by writing $\bar{\boldsymbol{x}}_k \overset{\text{abbr.}}{=} (\boldsymbol{x}_k, \boldsymbol{c}_k)$ such that the posterior PHD at t_{k-1}, the prior PHD at t_k, and the posterior PHD at t_k are, respectively,

$$\begin{aligned} v_{k-1}^+(\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1}) &\overset{\text{abbr.}}{=} v_{k-1}^+(\bar{\boldsymbol{x}}_{k-1}) \\ v_k^-(\boldsymbol{x}_k, \boldsymbol{c}_k) &\overset{\text{abbr.}}{=} v_k^-(\bar{\boldsymbol{x}}_k) \\ v_k^+(\boldsymbol{x}_k, \boldsymbol{c}_k) &\overset{\text{abbr.}}{=} v_k^+(\bar{\boldsymbol{x}}_k)\,. \end{aligned}$$

Then, in a manner similar to [87], the elements of the PHD that describe the surviving and spawned targets are given by

$$v_k^{(S)} = \int p_{S,k}(\bar{\boldsymbol{x}}_k) f(\bar{\boldsymbol{x}}_k|\bar{\boldsymbol{x}}_{k-1}) v_{k-1}^{+}(\bar{\boldsymbol{x}}_{k-1}) \mathrm{d}\bar{\boldsymbol{x}}_k \tag{4.2a}$$

$$v_k^{(\beta)} = \int \beta(\bar{\boldsymbol{x}}_k|\bar{\boldsymbol{x}}_{k-1}) v_{k-1}^{+}(\bar{\boldsymbol{x}}_{k-1}) \mathrm{d}\bar{\boldsymbol{x}}_k \,, \tag{4.2b}$$

where $p_{S,k}(\bar{\boldsymbol{x}}_k) \overset{\text{abbr.}}{=} p_{S,k}(\boldsymbol{x}_k, \boldsymbol{c}_k)$ is the state- and consider parameter-dependent probability of survival at time t_k, $f(\bar{\boldsymbol{x}}_k|\bar{\boldsymbol{x}}_{k-1}) \overset{\text{abbr.}}{=} f(\boldsymbol{x}_k, \boldsymbol{c}_k|\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1})$ is the joint transition density, and $\beta(\bar{\boldsymbol{x}}_k|\bar{\boldsymbol{x}}_{k-1}) \overset{\text{abbr.}}{=} \beta(\boldsymbol{x}_k, \boldsymbol{c}_k|\boldsymbol{x}_{k-1}, \boldsymbol{c}_{k-1})$ is the intensity of the spawning RFS given the previous state and consider parameters.

The PHD filter corrector can be written as the sum of two terms as

$$v_k^{+}(\bar{\boldsymbol{x}}_k) = v_k^{(D)} + v_k^{(\boldsymbol{z})} \,, \tag{4.3}$$

where $v_k^{(D)}$ is the contribution to the posterior intensity that accounts for missed detections of the targets and $v_k^{(\boldsymbol{z})}$ accounts for all of the data processed (i.e. all of the data contained in the RFS $\boldsymbol{Z}_k = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{M_k}\}$) at t_k. These updated intensity terms are given by

$$\begin{aligned} v_k^{(D)} &= [1 - p_{D,k}(\bar{\boldsymbol{x}}_k)] v_k^{-}(\bar{\boldsymbol{x}}_k) \\ v_k^{(\boldsymbol{z})} &= \sum_{\boldsymbol{z} \in \boldsymbol{Z}_k} \frac{p_{D,k}(\bar{\boldsymbol{x}}_k) g(\boldsymbol{z}|\bar{\boldsymbol{x}}_k) v_k^{-}(\bar{\boldsymbol{x}}_k)}{\kappa_k(\boldsymbol{z}) + \int p_{D,k}(\bar{\boldsymbol{\zeta}}_k) g(\boldsymbol{z}|\bar{\boldsymbol{\zeta}}_k) v_k^{-}(\bar{\boldsymbol{\zeta}}_k) \mathrm{d}\bar{\boldsymbol{\zeta}}_k} \,, \end{aligned}$$

where $p_{D,k}(\boldsymbol{x}_k, \boldsymbol{c}_k) \overset{\text{abbr.}}{=} p_{D,k}(\bar{\boldsymbol{x}}_k)$ is the state- and consider parameter-dependent probability of detection at time t_k, $g(\boldsymbol{z}|\boldsymbol{x}_k, \boldsymbol{c}_k) \overset{\text{abbr.}}{=} g(\boldsymbol{z}|\bar{\boldsymbol{x}}_k)$ is the likelihood function, and $\kappa_k(\boldsymbol{z})$ is the intensity of the clutter RFS at time t_k. A major advantage to this approach over many multitarget tracking mechanisms is that data association, often a very costly and heuristic-based operation, is built directly into the model-based formulation of the PHD filter. In fact, the data association happens directly on the state space through $g(\boldsymbol{z}|\bar{\boldsymbol{x}}_k)$, a trait that produces a very efficient filter.

The estimated number of objects tracked by the PHD filter, i.e. its cardinality estimate is defined as the integral of the intensity function over the entire multitarget space; that is

$$\hat{N}_k = \int v_k(\bar{\boldsymbol{x}}_k)\mathrm{d}\bar{\boldsymbol{x}}_k \,.$$

The estimated number of tracked objects can be computed at any time using either the prior or posterior intensity. Additionally, an estimate of the number of targets within a specific region can be found by restricting the domain of integration to that specific region.

The relationships defining the predictor and corrector of the PHD filter offer a recursion that depends on both the target states and the consider parameters, but the desired algorithm must *consider* the error in these parameters and not update any type of estimate in the consider parameters. This is to be accomplished in much the same way as was described for single target Bayesian consider filtering in Section 3.3.[6] In particular, a GM approximation to the joint intensity is applied, and the consider parameters are marginalized out from the resulting expressions. In a manner inspired by the way Vo produced a GM PHD filter in [69], here, a GM consider PHD filter is developed using the results of Section 3.3, leveraging the new, generalized forms of Lemmas 3.3 and 3.4.

The modeling decisions of the Bayesian consider results in Section 3.3 are utilized in the following, such as the definition of transition kernels for the state and consider parameters, so these models will be referenced directly and repeated here when necessary.

4.2.1.1. Predictor. Equations (3.42) and (3.43), repeated here as

$$f(\boldsymbol{x}_k|\boldsymbol{x}_{k-1},\boldsymbol{c}_{k-1}) = p_g(\boldsymbol{x}_k\,;\,\boldsymbol{F}_{x,k-1}\boldsymbol{x}_{k-1}+\boldsymbol{F}_{c,k-1}\boldsymbol{c}_{k-1},\boldsymbol{F}_{w,k-1}\boldsymbol{P}_{ww,k-1}\boldsymbol{F}_{w,k-1}^T)$$
$$f(\boldsymbol{c}_k|\boldsymbol{c}_{k-1}) = p_g(\boldsymbol{c}_k\,;\,\boldsymbol{G}_{c,k-1}\boldsymbol{c}_{k-1},\boldsymbol{G}_{u,k-1}\boldsymbol{P}_{uu,k-1}\boldsymbol{G}_{u,k-1}^T)\,,$$

[6] Indeed, this was a principal motivation for the developments of that section, as the PHD filter is interpreted here as a Bayesian, albeit multitarget, consider filter

describe the stochastic, temporal evolution of a target state and the consider parameters, respectively. Assuming these transition densities are independent, their joint transition density can be written as[7]

$$f(\bar{\boldsymbol{x}}_k|\bar{\boldsymbol{x}}_{k-1}) = p_g\left(\begin{bmatrix}\boldsymbol{x}_k\\ \boldsymbol{c}_k\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{F}_{x,k-1} & \boldsymbol{F}_{c,k-1}\\ \boldsymbol{0}_{n_c\times n_x} & \boldsymbol{G}_{c,k-1}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_{k-1}\\ \boldsymbol{c}_{k-1}\end{bmatrix},\begin{bmatrix}\boldsymbol{P}^*_{ww,k-1} & \boldsymbol{0}_{n_x\times n_c}\\ \boldsymbol{0}_{n_c\times n_x} & \boldsymbol{P}^*_{uu,k-1}\end{bmatrix}\right),$$

where

$$\begin{aligned}\boldsymbol{P}^*_{ww,k-1} &= \boldsymbol{F}_{w,k-1}\boldsymbol{P}_{ww,k-1}\boldsymbol{F}^T_{w,k-1}\\ \boldsymbol{P}^*_{uu,k-1} &= \boldsymbol{G}_{u,k-1}\boldsymbol{P}_{uu,k-1}\boldsymbol{G}^T_{u,k-1}\,.\end{aligned}$$

Let the PHDs of the spawning and birth RFSs be GMs of the forms

$$\begin{aligned}\beta(\bar{\boldsymbol{x}}_k|\bar{\boldsymbol{x}}_{k-1}) &= \sum_{\ell=1}^{L^{(\beta)}_{k-1}} w^{(\beta)}_{\ell,k-1}p_g\left(\begin{bmatrix}\boldsymbol{x}_k\\ \boldsymbol{c}_k\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{F}^{(\beta)}_{x,k-1} & \boldsymbol{F}^{(\beta)}_{c,k-1}\\ \boldsymbol{0} & \boldsymbol{G}_{c,k-1}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_{k-1}\\ \boldsymbol{c}_{k-1}\end{bmatrix},\begin{bmatrix}\boldsymbol{P}^{(\beta)}_{ww,\ell,k-1} & \boldsymbol{0}_{n_x\times n_c}\\ \boldsymbol{0}_{n_c\times n_x} & \boldsymbol{P}^*_{uu,k-1}\end{bmatrix}\right)\\ \gamma(\bar{\boldsymbol{x}}_k) &= \sum_{\ell=1}^{L^{(\gamma)}_{k}} w^{(\gamma)}_{\ell,k}p_g\left(\begin{bmatrix}\boldsymbol{x}_{k-1}\\ \boldsymbol{c}_{k-1}\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{m}^{(\gamma)}_{x,\ell,k}\\ \boldsymbol{m}^-_{c,k}\end{bmatrix},\begin{bmatrix}\boldsymbol{P}^{(\gamma)}_{xx,\ell,k} & \boldsymbol{P}^{(\gamma)}_{xc,\ell,k}\\ (\boldsymbol{P}^{(\gamma)}_{xc,\ell,k})^T & \boldsymbol{P}^-_{cc,k}\end{bmatrix}\right)\end{aligned}$$

respectively, where $w^{(\beta)}_{\ell,k-1}, \boldsymbol{F}^{(\beta)}_{x,k-1}$, $\boldsymbol{F}^{(\beta)}_{c,k-1}$, and $\boldsymbol{P}^{(\beta)}_{ww,\ell,k-1}$ govern the shape of the PHD of the spawned targets and $w^{(\gamma)}_{\ell,k}, \boldsymbol{m}^{(\gamma)}_{x,\ell,k}$, $\boldsymbol{P}^{(\gamma)}_{xx,\ell,k}$, and $\boldsymbol{P}^{(\gamma)}_{xc,\ell,k}$ describe the shape of the intensity of the born targets. Note the appearance of the $\boldsymbol{P}^{(\gamma)}_{xc,\ell,k}$ terms. Again, these terms capture correlations between the target states and the consider parameters in the GM representation of the multitarget intensity. These terms are vital to the success of a consider filter implementation and contain useful correlations that update the filter's knowledge of the PHD approximation. Note, too, that the consider parameter vector is the same as before, even for the spawned and born targets. This is because these consider parameters are present in the systems into which these new targets appear and are not target specific. The use of different consider parameters that require specific knowledge of target identity, something

[7]Note that the dynamics matrices $\boldsymbol{F}_{\alpha,k-1}$ can be GM-component dependent, i.e. $\boldsymbol{F}_{\alpha,k-1} = \boldsymbol{F}_{\alpha,\ell,k-1}$, and, indeed, this is the case when linearization is used to approximate system nonlinearities as discussed at length in this dissertation. This dependency will be omitted, however, to simplify the resulting expressions. A reader should simply understand that the ℓ^{th} Jacobian is evaluated at the ℓ^{th} component's mean, and similar logic extends to quadrature schemes.

that the PHD filter, as formulated, makes no attempt to estimate, cannot be employed. Additionally, assume the posterior intensity at time t_{k-1} is described by a GM of the form

$$v_{k-1}^{+}(\bar{\boldsymbol{x}}_{k-1}) = \sum_{\ell=1}^{L_{k-1}^{+}} w_{\ell,k-1}^{+} p_g \left(\begin{bmatrix} \boldsymbol{x}_{k-1} \\ \boldsymbol{c}_{k-1} \end{bmatrix} ; \begin{bmatrix} \boldsymbol{m}_{x,\ell,k-1}^{+} \\ \boldsymbol{m}_{c,k-1}^{+} \end{bmatrix}, \begin{bmatrix} \boldsymbol{P}_{xx,\ell,k-1}^{+} & \boldsymbol{P}_{xc,\ell,k-1}^{+} \\ (\boldsymbol{P}_{xc,\ell,k-1}^{+})^T & \boldsymbol{P}_{cc,k-1}^{+} \end{bmatrix} \right) .$$

Finally, assume

$$p_{S,k}(\boldsymbol{x}_k, \boldsymbol{c}_k) = p_{S,k} \, ,$$

i.e. that the probability of survival is independent of the state and consider parameters.[8]

Then, under these assumptions, an application of Lemma 3.3 to the integral-products in Eqs. (4.2) permits them to be written as

$$\begin{aligned} v_k^{(S)} &= \sum_{\ell=1}^{L_{k-1}^{+}} w_{\ell,k}^{(S)} p_g \left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{c}_k \end{bmatrix} ; \begin{bmatrix} \boldsymbol{m}_{x,\ell,k}^{(S)} \\ \boldsymbol{m}_{c,k}^{-} \end{bmatrix}, \begin{bmatrix} \boldsymbol{P}_{xx,\ell,k}^{(S)} & \boldsymbol{P}_{xc,\ell,k}^{(S)} \\ (\boldsymbol{P}_{xc,\ell,k}^{(S)})^T & \boldsymbol{P}_{cc,k}^{-} \end{bmatrix} \right) \\ v_k^{(\beta)} &= \sum_{\ell=1}^{L_{k-1}^{+}} \sum_{j=1}^{L_{k-1}^{(\beta)}} w_{\ell,k-1}^{+} w_{j,k-1}^{(\beta)} p_g \left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{c}_k \end{bmatrix} ; \begin{bmatrix} \boldsymbol{m}_{x,\ell,k}^{(\beta)} \\ \boldsymbol{m}_{c,k}^{-} \end{bmatrix}, \begin{bmatrix} \boldsymbol{P}_{xx,(\ell,j),k}^{(\beta)} & \boldsymbol{P}_{xc,\ell,k}^{(\beta)} \\ (\boldsymbol{P}_{xc,\ell,k}^{(\beta)})^T & \boldsymbol{P}_{cc,k}^{-} \end{bmatrix} \right) \end{aligned}$$

where

$$\begin{aligned} w_{\ell,k}^{(S)} &= p_{S,k} w_{\ell,k-1}^{+} \\ \boldsymbol{m}_{x,\ell,k}^{(S)} &= \boldsymbol{F}_{x,k-1} \boldsymbol{m}_{x,\ell,k-1}^{+} + \boldsymbol{F}_{c,k-1} \boldsymbol{m}_{c,k-1}^{+} \\ \boldsymbol{m}_{c,k}^{-} &= \boldsymbol{G}_{c,k-1} \boldsymbol{m}_{c,k-1}^{+} \\ \boldsymbol{P}_{xx,\ell,k}^{(S)} &= \boldsymbol{F}_{x,k-1} \boldsymbol{P}_{xx,\ell,k-1}^{+} \boldsymbol{F}_{x,k-1}^{T} + \boldsymbol{F}_{c,k-1} \boldsymbol{P}_{cc,k-1}^{+} \boldsymbol{F}_{c,k-1}^{T} + \boldsymbol{F}_{w,k-1} \boldsymbol{P}_{ww,\ell,k-1}^{+} \boldsymbol{F}_{w,k-1}^{T} \\ &\quad + \boldsymbol{F}_{x,k-1} \boldsymbol{P}_{xc,\ell,k-1}^{+} \boldsymbol{F}_{c,k-1}^{T} + \boldsymbol{F}_{c,k-1} (\boldsymbol{P}_{xc,\ell,k-1}^{+})^T \boldsymbol{F}_{x,k-1}^{T} \\ \boldsymbol{P}_{cc,k}^{-} &= \boldsymbol{G}_{c,k-1} \boldsymbol{P}_{cc,k-1}^{+} \boldsymbol{G}_{c,k-1}^{T} + \boldsymbol{G}_{u,k-1} \boldsymbol{P}_{uu,k-1}^{+} \boldsymbol{G}_{u,k-1}^{T} \\ \boldsymbol{P}_{xc,\ell,k}^{(S)} &= \boldsymbol{F}_{x,k-1} \boldsymbol{P}_{xc,\ell,k-1}^{+} \boldsymbol{G}_{c,k-1}^{T} + \boldsymbol{F}_{c,k-1} \boldsymbol{P}_{cc,k-1}^{+} \boldsymbol{G}_{c,k-1}^{T} \end{aligned}$$

[8] A useful approximation to relax this somewhat is to make the probability of survival a GM component-dependent function and make the zeroth-order approximation $p_{S,k} = p_{S,k}(\boldsymbol{m}_{x,\ell,k-1}^{+})$. Modification of the forthcoming equations to accommodate this follows logically.

and

$$\begin{aligned}
\boldsymbol{m}_{\ell,k}^{(\beta)} &= \boldsymbol{F}_{x,k-1}^{(\beta)}\boldsymbol{m}_{x,\ell,k-1}^{+} + \boldsymbol{F}_{c,k-1}^{(\beta)}\boldsymbol{m}_{c,k-1}^{+} \\
\boldsymbol{P}_{xx,(\ell,j),k}^{(\beta)} &= \boldsymbol{F}_{x,k-1}^{(\beta)}\boldsymbol{P}_{xx,\ell,k-1}^{+}(\boldsymbol{F}_{x,k-1}^{(\beta)})^{T} + \boldsymbol{F}_{c,k-1}^{(\beta)}\boldsymbol{P}_{cc,k-1}^{+}(\boldsymbol{F}_{c,k-1}^{(\beta)})^{T} + \boldsymbol{P}_{ww,j,k-1}^{(\beta)} \\
&\quad + \boldsymbol{F}_{c,k-1}^{(\beta)}(\boldsymbol{P}_{xc,\ell,k-1}^{+})^{T}(\boldsymbol{F}_{x,k-1}^{(\beta)})^{T} + \boldsymbol{F}_{x,k-1}^{(\beta)}\boldsymbol{P}_{xc,\ell,k-1}^{+}(\boldsymbol{F}_{c,k-1}^{(\beta)})^{T} \\
\boldsymbol{P}_{xc,\ell,k}^{(\beta)} &= \boldsymbol{F}_{x,k-1}^{(\beta)}\boldsymbol{P}_{xc,\ell,k-1}^{+}\boldsymbol{G}_{c,k-1}^{T} + \boldsymbol{F}_{c,k-1}^{(\beta)}\boldsymbol{P}_{cc,k-1}^{+}\boldsymbol{G}_{c,k-1}^{T}\,.
\end{aligned}$$

Then, the three summed terms in Eq. (4.1) are given by GMs, and accordingly $v_k^-(\bar{\boldsymbol{x}}_k)$ is also a GM. That is,

$$v_k^-(\bar{\boldsymbol{x}}_k) = \sum_{\ell=1}^{L_k^-} w_{\ell,k}^- p_g\left(\begin{bmatrix}\boldsymbol{x}_k \\ \boldsymbol{c}_k\end{bmatrix};\begin{bmatrix}\boldsymbol{m}_{x,\ell,k}^- \\ \boldsymbol{m}_{c,k}^-\end{bmatrix},\begin{bmatrix}\boldsymbol{P}_{xx,\ell,k}^- & \boldsymbol{P}_{xc,\ell,k}^- \\ (\boldsymbol{P}_{xc,\ell,k}^-)^T & \boldsymbol{P}_{cc,k}^-\end{bmatrix}\right) \tag{4.6}$$

following the collection of terms in Eq. (4.1). The PHD $v_k^-(\bar{\boldsymbol{x}}_k)$ contains the components of the surviving, spawned, and born GMs, and this can be a point of puzzlement for those implementing such filters for the first time. A sum of GMs is itself a GM, and in implementation, one simply takes the three GMs of Eq. (4.1) and combines them into a single GM with $L_k^- = L_{k-1}^+(1 + L_{k-1}^{(\beta)}) + L_k^{(\gamma)}$ components. Note that this implies that the number of GM components grows through time without bound, and this undesirable element will be discussed later.

If desired, the prior marginal intensity of $\boldsymbol{x}_k$, i.e. without dependence on the parameters, can be obtained by factoring each GM component as shown in Section 3.3 and integrating over $\boldsymbol{c}_k$.

4.2.1.2. Corrector. Given this GM description of the prior intensity, the corrector stage of the recursion aims to incorporate information from the measurement RFS $\boldsymbol{Z}_k$ using Eq. (4.3). In a similar manner to the predictor step in the previous section, it must be assumed that the probability of detection is independent of state or consider parameter,

meaning that[9]

$$p_{D,k}(\boldsymbol{x}_k, \boldsymbol{c}_k) = p_{D,k}\,.$$

Also, it will be assumed that the likelihood function is given by Eq. (3.44). Equation (4.3) can then be written as

$$\begin{aligned} v_k^+(\bar{\boldsymbol{x}}_k) = {} & [1 - p_{D,k}] v_k^-(\bar{\boldsymbol{x}}_k) \\ & + \sum_{\boldsymbol{z} \in \boldsymbol{Z}_k} \frac{p_{D,k} p_g(\boldsymbol{z}\,;\, \boldsymbol{H}_{x,k}\boldsymbol{x}_k + \boldsymbol{H}_{c,k}\boldsymbol{c}_k, \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}_{v,k}^T) v_k^-(\bar{\boldsymbol{x}}_k)}{\kappa_k(\boldsymbol{z}_k) + \int p_{D,k} p_g(\boldsymbol{z}\,;\, \boldsymbol{H}_{x,k}\boldsymbol{x}_k + \boldsymbol{H}_{c,k}\boldsymbol{c}_k, \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}_{v,k}^T) v_k^-(\bar{\boldsymbol{x}}_k)}\,, \end{aligned}$$

where $v_k^-(\bar{\boldsymbol{x}}_k)$ is the GM given by Eq. (4.6). Recalling the abbreviations introduced previously (omitting the explicit dependence on the consider parameters, i.e. $v_k^-(\bar{\boldsymbol{x}}_k) = v_k^-(\boldsymbol{x}_k, \boldsymbol{c}_k)$), the above expression is integrated over $\boldsymbol{c}_k$ to obtain the posterior intensity only in $\boldsymbol{x}_k$. Performing this integration, and applying Lemmas 3.3 and 3.4, the posterior GM intensity, now without dependence upon $\boldsymbol{c}_k$, can be written as

$$v_k^+(\boldsymbol{x}_k) = [1 - p_{D,k}] \sum_{\ell=1}^{L_k^-} p_g(\boldsymbol{x}_k\,;\, \boldsymbol{m}_{x,\ell,k}^-, \boldsymbol{P}_{xx,\ell,k}^-) + \sum_{\boldsymbol{z} \in \boldsymbol{Z}_k} \sum_{\ell=1}^{L_k^-} \hat{w}_{\ell,k} p_g(\boldsymbol{x}_k\,;\, \hat{\boldsymbol{m}}_{x,\ell,k}, \hat{\boldsymbol{P}}_{xx,\ell,k})\,,$$

where

$$\begin{aligned} \hat{w}_{\ell,k} &= \frac{p_{D,k} w_{\ell,k}^- q_{\ell,k}(\boldsymbol{z})}{\kappa_k(\boldsymbol{z}) + \sum_{j=1}^{L_k^-} p_{D,k} w_{j,k}^- q_{j,k}(\boldsymbol{z})} \\ q_{\ell,k}(\boldsymbol{z}) &= p_g(\boldsymbol{z}\,;\, \boldsymbol{H}_{x,k}\boldsymbol{m}_{x,\ell,k}^- + \boldsymbol{H}_{c,k}\boldsymbol{m}_{c,k}^-, \boldsymbol{P}_{zz,\ell,k}^-) \\ \hat{\boldsymbol{m}}_{x,\ell,k} &= \boldsymbol{m}_{x,\ell,k}^- + \boldsymbol{K}_{x,\ell,k}(\boldsymbol{z}_k - \boldsymbol{H}_{x,k}\boldsymbol{m}_{x,\ell,k}^- - \boldsymbol{H}_{c,k}\boldsymbol{m}_{c,k}^-) \\ \hat{\boldsymbol{P}}_{xx,\ell,k} &= (\boldsymbol{I}_{n_x \times n_x} - \boldsymbol{K}_{x,\ell,k}\boldsymbol{H}_{x,k})\boldsymbol{P}_{xx,\ell,k}^- - \boldsymbol{K}_{x,\ell,k}\boldsymbol{H}_{c,k}(\boldsymbol{P}_{xc,\ell,k}^-)^T \\ \hat{\boldsymbol{P}}_{xc,\ell,k} &= (\boldsymbol{I}_{n_x \times n_x} - \boldsymbol{K}_{x,\ell,k}\boldsymbol{H}_{x,k})\boldsymbol{P}_{xc,\ell,k}^- - \boldsymbol{K}_{x,\ell,k}\boldsymbol{H}_{c,k}\boldsymbol{P}_{cc,k}^- \\ \boldsymbol{K}_{x,\ell,k} &= \boldsymbol{P}_{xz,\ell,k}^-(\boldsymbol{P}_{zz,\ell,k}^-)^{-1} \end{aligned}$$

[9] Again, a very powerful (and practically necessary) modification of this procedure is the approximation that the probability of detection is functionally dependent on the mean of that component, i.e. $p_{D,k} = p_{D,k}(\boldsymbol{m}_{x,\ell,k}^-)$. This is the manner with which targets leaving and entering the field of view can be accounted for, such as setting $p_{D,k}$ to some nominal value if a component is expected to be in the field of view and zero otherwise.

$$\begin{aligned}
\boldsymbol{P}^{-}_{xz,\ell,k} &= \boldsymbol{P}^{-}_{xx,\ell,k}\boldsymbol{H}^{T}_{x,k} + \boldsymbol{P}^{-}_{xc,\ell,k}\boldsymbol{H}^{T}_{c,k} \\
\boldsymbol{P}^{-}_{zz,\ell,k} &= \boldsymbol{H}_{x,k}\boldsymbol{P}^{-}_{xx,\ell,k}\boldsymbol{H}^{T}_{x,k} + \boldsymbol{H}_{c,k}\boldsymbol{P}^{-}_{cc,k}\boldsymbol{H}^{T}_{c,k} + \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}^{T}_{v,k} \\
&\quad + \boldsymbol{H}_{x,k}\boldsymbol{P}^{-}_{xc,\ell,k}\boldsymbol{H}^{T}_{c,k} + \boldsymbol{H}_{c,k}(\boldsymbol{P}^{-}_{xc,\ell,k})^{T}\boldsymbol{H}^{T}_{x,k}\,.
\end{aligned}$$

Consistent with the theme of a consider filter, no update is performed on $\boldsymbol{m}^{-}_{c,k}$ or $\boldsymbol{P}^{-}_{cc,k}$, i.e.

$$\begin{aligned}
\boldsymbol{m}^{+}_{c,k} &= \boldsymbol{m}^{-}_{c,k} \\
\boldsymbol{P}^{+}_{cc,k} &= \boldsymbol{P}^{-}_{cc,k}\,.
\end{aligned}$$

Note, however, that the prior correlations, as well as the measurement-updated correlations, are maintained, and that they contribute significant differences to a non-consider formulation. This allows the filter to maintain knowledge of correlations between the state and uncertain parameters without requiring the parameters to be estimated. Furthermore, while the expressions are more complicated, each matrix operation is of a lower dimension than a "traditional" GM PHD filter where the uncertain parameters are estimated. Depending on the application, particularly parameter observability and computational complexity requirements, this may be a desired trait.

The posterior intensity (in the state and only the state) is then the sum of two GMs, which is itself a GM, allowing it to be written as

$$v^{+}_{k}(\boldsymbol{x}_k) = \sum_{\ell=1}^{L^{+}_{k}} w^{+}_{\ell,k}\, p_g(\boldsymbol{x}_k\,;\, \boldsymbol{m}^{+}_{x,\ell,k}, \boldsymbol{P}^{+}_{xx,\ell,k})\,,$$

but it should be noted that the filter still maintains the component-wise posterior correlations matrix in its memory for use in later recursion cycles. Observation of these expressions indicates that $L^{+}_{k} = (1 + |\boldsymbol{Z}_k|)L^{-}_{k}$. That is, as with the predictor stage, the number of components grows through time.

State estimates are taken to be the modes (or peaks) within the GM intensity, and the computational complexity reduction techniques for the GM PHD filter, such as pruning and merging, apply directly to the GM consider PHD filter [69]. These techniques are of

paramount importance to obtaining a practical application of this filter and should be used at all times. These techniques are not the innovations of this dissertation and are specific details are therefore omitted here, but this is briefly discussed in the following section.

Finally, note that, in contrast to the consider GM filter of Section 3.3.3 where the sum of all GM weights equals one, here the sum of the GM weights at any time produces the total cardinality estimate $\hat{N}_k$ at that time. Proof of this is trivial if one recalls that the integral of an intensity yields the anticipated number of targets.

Under certain assumptions, the GM consider PHD filter shares algorithmic equivalence to the methods inspiring this approach. This serves as a useful method to perform a sanity check on the proposed GM consider PHD filter.

Remark 4.2. *In the case where the uncertainty in the consider parameter vector vanishes, along with its correlation to the target states, or where the consider parameters are absent/neglected, the GM consider PHD filter is algorithmically equivalent to the GM PHD filter of [69].*

Remark 4.3. *In the case of tracking a single target, where probability of detection is one, survival is guaranteed, birth and spawning are neglected, and the rate of clutter returns is zero, the GM consider PHD filter is algorithmically equivalent to the GM consider filter (from Section 3.3.3).*

Remark 4.4. *Under the same conditions as Remark 4.3 and the additional constraint that the intensity is characterized by a single Gaussian, the GM consider PHD filter is algorithmically equivalent to the consider filter of Section 3.1.*

4.2.2. Brief Remarks on GM Implementations. A well-known complication of most, if not all, multitarget tracking algorithms is an increase in complexity through time, and in GM implementations of filters such as the PHD filter, this complexity manifests itself as a steadily growing number of GM components. While this poses no theoretical challenges, it substantially taxes a computer implementation, ultimately causing the filter to grind to a halt as the computer is forced to perform operations on each of a huge number of components. Furthermore, in the case that many GM components represent the same

target, state estimation (nearly always a mode finding problem) is drastically degraded in both efficiency and accuracy. However, many methods have been developed to stave off the exponential growth of mixands, and the two most common techniques are known as pruning and merging, respectively.

Pruning is quite simple in concept and execution: given a collection of GM components, define a threshold below which it is said that components are insignificant (some small number, such as 1×10^{-10}), and then remove any components with weights below that threshold from the GM. It is important that, if the GM is representing a PHD, the resulting pruned weights are not normalized as one would do if representing a pdf.

Merging is a more complex procedure, but it serves as the key to a successful PHD implementation. To perform merging, first compute the pairwise Mahalanobis distances between all of the components in the GM. Since the Mahalanobis distance between a sample from a normal density is known to be χ^2 distributed, a user can define an agreement threshold between components and using a standard χ^2 lookup table to determine if components statistically agree. For example, a user can choose to merge any components that agree according to a 95% χ^2 test. A χ^2 table lookup is performed for a 95% agreement for the degrees of freedom of the problem (i.e. dimension of the GM components), and any components with Mahalanobis distance below the corresponding table value are merged using the method of moments. This winds up being computationally burdensome for many-component GMs, due to needing to compute the pairwise distances between all the components, but it is crucial to a successful implementation, principally for two reasons:

- While it is a relatively costly procedure for many-component GMs, electing to skip merging entirely results in an even larger GM. Bearing the computational cost earlier saves an extraordinary amount of computational cost later.

- Most GM PHD state estimation techniques rely on finding the highest weighted component in a given neighborhood. Merging takes a cluster of components in a given neighborhood and replaces them by a single component to approximate that cluster. If state estimation is attempted on the cluster, there may be no way to say which

component in the cluster should be accepted as a state estimate. Once merged, however, state estimation is drastically simplified: the state estimate is clearly the single component.

Since the cost associated with merging scales with the number of components in the GM to be merged, it can be tempting to perform pruning first and then merge the resulting GM. It is emphatically stated that this is poor reasoning, and, instead, merging should *always* be performed *before* pruning. To see why, consider an illustrative case where a single target is represented by a million equally weighted components. If pruning is performed before merging, that target is likely to be deleted entirely from the GM, but if merging is performed, those million components will likely be reduced to a handful of Gaussian components, or perhaps a single Gaussian. This is an extreme example, but as scenario complexity increases, these types of situations begin to arise. A compromise may be to conduct a softened "pre-merging prune" where pruning is done to remove all components with weights that are very nearly zero, such as being below a much smaller number like 1×10^{-30}, and then perform merging and pruning as normal.

For specific details on how to implement pruning and merging, see [69].

4.2.3. Other Multitarget Consider Filters. With what has been presented up to this point, it stands mostly as an algebraic exercise to derive GM consider formulations of other popular multitarget filters, such as the CPHD filter [70, 88], the multitarget multi-Bernoulli filter [71], or the δ-GLMB filter [72, 73]. Derivations of GM formulations of these filters, as presented in the referenced literature, rely upon the well-known Lemmas 3.1 and 3.2 to establish the desired results. A principal focus of this dissertation is the use of their generalizations that were developed in Section 3, Lemmas 3.3 and 3.4, to enable Bayesian consider filtering. If one follows the same procedures as outlined in Section 3.3, as well as those outlined just before this, use of these lemmas handily produces the desired multitarget consider filters. In fact, if one "squints" at the standard (i.e. non-consider) formulations of these filters in literature long enough, it becomes rather clear how to immediately obtain a consider formulation. This largely motivated the progression of deriving a Gaussian result (Section 3.3.2), leading to a non-Gaussian/GM result (Section 3.3.3), and,

ultimately, a multitarget consider PHD result (Section 4.2.1). While these are very different tools, they rely on the same principles to produce practical results, and, as such, explicit derivation is omitted here and is left to an interested reader.

4.2.4. Evaluating Multitarget Filters: The OSPA Metric. In the tracking problem, the concept of the miss-distance is often taken for granted in the single-target sense, a performance metric often trivially computed as error, but in the multitarget realm, a reliable metric is much less straightforward. In this case, a metric that describes the difference between two RFSs needs to be quantified. This metric needs to contain consistent information about both the errors in localization and cardinality between the two sets and it needs to be able to be interpreted in some way that makes sense to the problem (usually a physical interpretation is desired). One such metric that has proven immensely useful in multitarget tracking is the optimal subpattern assignment (OSPA) metric [103]. The metric, given two arbitrary RFSs $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ and $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$ of cardinality m and n respectively, is written as

$$\bar{d}_p^{(c)}(\boldsymbol{X}, \boldsymbol{Y}) = \left[\frac{1}{n}\left(\min_{\pi \in \Pi_n} \sum_{i=1}^{m} d^{(c)}(\boldsymbol{x}_i, \boldsymbol{y}_{\pi(i)})^p + c^p(n-m)\right)\right]^{1/p} \tag{4.8}$$

where Π_n is the set of all permutations on $\{1, 2, \ldots, n\}$ between the sets for $m \leq n$, $d^{(c)}(\boldsymbol{x}_i, \boldsymbol{y}_{\pi(i)})$ is some metric distance between the points $\boldsymbol{x}_i$ and $\boldsymbol{y}_{\pi(i)}$ (usually taken to be the standard Euclidean norm), c is a maximum distance cutoff parameter between these points, and p is the order of the metric. The parameter c can be interpreted as a way of selecting "how far away" two points have to be to be considered a false association or missing estimate, and p can be interpreted in much the same way as p^{th}-order norms, where higher values of p become less and less forgiving to outliers.

For the case where $m > n$, the OSPA distance is simply computed via $\bar{d}_p^{(c)}(\boldsymbol{Y}, \boldsymbol{X})$. Additionally, the metric can be rewritten in such a way that the errors in localization and cardinality are expressed separately, given by

$$d_{\text{loc}}(\boldsymbol{X}, \boldsymbol{Y}) = \left[\frac{1}{n}\left(\min_{\pi \in \Pi_n} \sum_{i=1}^{m} d^{(c)}(\boldsymbol{x}_i, \boldsymbol{y}_{\pi(i)})^p\right)\right]^{1/p} \tag{4.9}$$

$$d_{\text{card}}(\boldsymbol{X},\boldsymbol{Y}) = \left[\frac{c^p\,(n-m)}{n}\right]^{1/p}. \tag{4.10}$$

Using these components of OSPA is especially useful because it allows a deeper understanding of the accuracy in the multitarget state estimates independent of cardinality errors. Sometimes it can be useful to note that, if $p = 1$, $\bar{d}_p^{(c)}(\boldsymbol{X},\boldsymbol{Y}) = d_{\text{loc}}(\boldsymbol{X},\boldsymbol{Y}) + d_{\text{card}}(\boldsymbol{X},\boldsymbol{Y})$. That is, in the case that the selected distance measure is the standard error, the total OSPA is the sum of the localization and cardinality components. It is most common to set $p = 2$ since it tends to produce smoother curves and, therefore, assists in interpreting results [103], but, sometimes, the interpretation of "average per target error" for the $p = 1$ case can produce useful insight into simpler problems.

Computing the set of all pairwise permutations between the elements of two finite sets is a classic example of the assignment problem and remains a task that is inherently of appreciable computational burden, particularly as the cardinality of the random finite sets of interest grows. The Hungarian algorithm was developed for the solution of such assignment problems and greatly reduces the time required to compute the OSPA metric [104].

4.2.5. Numerical Example. The following example, while potentially unrealistic in some regards, is designed to clearly illustrate the impact that uncertain model parameters can have on a multitarget estimation problem. The design of the problem is motivated by two observations:

1. problems that arguably benefit the most from consider analysis are those of high-dimensional, real-world problems that are impractical to demonstrate compactly in a small-scale simulation demonstration, and

2. consider analysis offers tremendous benefits in cases where parameters are "weakly" observable, which are parameters that are observable but take so long to accumulate useful correlations that estimating their value is perhaps a waste of computational resources.

Accordingly, the following simulation is designed to illustrate a case where a consider analysis offers advantages over a filter that instead estimates the model parameters. In this case, the derived GM consider PHD filter is compared to the standard GM PHD filter of [69] that would estimate parameters as part of the target states rather than consider them.

Consider a vehicle in low Earth orbit, such as the International Space Station, ejecting three identical small satellites, such as cubesats, from a cargo hatch. In order to ensure no harm comes to the station, onboard sensors track the cubesats as they drift away due to the effects of relative dynamics, zonal harmonics, and drag. Take the state of the station at time t_k to be given by $\boldsymbol{x}_{s,k}$, and each target state to be of inertial position and velocity such that

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r}^T & \boldsymbol{v}^T \end{bmatrix}^T = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T .$$

The station and each target evolve under the influence of point mass and J_2 gravitational and spherical drag accelerations, such that the equations of motion obey

$$\ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{a}_{J_2} + \boldsymbol{a}_{\text{drag}} ,$$

where

$$\boldsymbol{a}_{J_2} = -\frac{3}{2}J_2\left(\frac{\mu R_e^2}{r^3}\right)\begin{bmatrix} \left(1-5\left(\frac{z}{r}\right)^2\right)\frac{x}{r} \\ \left(1-5\left(\frac{z}{r}\right)^2\right)\frac{y}{r} \\ \left(3-5\left(\frac{z}{r}\right)^2\right)\frac{z}{r} \end{bmatrix}$$
$$\boldsymbol{a}_{\text{drag}} = -\frac{1}{2}C_D\frac{A}{m}\rho v_r^2\left(\frac{\boldsymbol{v}_r}{v_r}\right) ,$$

μ is the gravitational parameter, J_2 is the Earth oblateness gravitational factor, R_e is the equatorial radius of the Earth, C_D is the drag coefficient, A is the surface area exposed to atmosphere inducing drag, m is the mass of the body, ρ is the atmospheric density, and $\boldsymbol{v}_r$ is the relative wind. The density is taken to be exponentially defined with a sinusoidal term

of the form

$$\rho(h,t) = \rho_0 \exp\left\{-\frac{h-h_0}{H}\right\} + a\sin(\omega t + \psi)\,,$$

where the sinusoidal term is (arbitrarily) designed to account for periodic atmospheric density variations as the spacecraft travel above different portions of the Earth. Pragmatically, this is introduced to add more hard-to-estimate parameters to the problem (motivated by the previously mentioned observations). For this model, the values $h_0 = 400$ km, $H = 58.515$ km, and $\rho_0 = 3.725 \times 10^{-12}$ kg/m^2 are used.

The station is taken to be in a circular, 400 km, 50° inclined orbit with initial radius r_0 such that its initial state is given by

$$\boldsymbol{x}_{s,0} = \begin{bmatrix}6778136.3 & 0 & 0 & 0 & 4929.25 & 5874.46\end{bmatrix}^T$$

in m and m/s. Each satellite that is ejected from the station is taken to be collocated with the station but is instantaneously influenced by a Δv of 1 m/s, each in a random direction. The initial GM representation of the target PHD is constructed to initialize in such a way that each target contributes to the GM representation of the PHD in a unique way. First, each of the 3 target's contributions to the intensity is taken to be a single Gaussian component (each with weight 1) with covariance described by

$$\boldsymbol{P}_{xx,\ell,0} = \mathrm{diag}\{\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_{\dot{x}}^2, \sigma_{\dot{y}}^2, \sigma_{\dot{z}}^2\}\,,$$

where $\sigma_x^2, \sigma_y^2, \sigma_z^2 \sim \mathcal{U}(3,5)$ in m^2, $\sigma_{\dot{x}}^2, \sigma_{\dot{y}}^2, \sigma_{\dot{z}}^2 \sim \mathcal{U}(0.003, 0.005)$ in (m/s)^2, and $\mathcal{U}(a,b)$ is the uniform distribution on (a,b). Each $\boldsymbol{P}_{xx,\ell,0}$ is then rotated by a random angle to produce correlations within the state. An initial mean is obtained for each target component, for target j and true target state $\boldsymbol{x}^{(j)}$, by drawing $\boldsymbol{m}_{x,j,0} \sim \mathcal{N}(\boldsymbol{x}^{(j)}, \boldsymbol{P}_{xx,j,0})$, where $\mathcal{N}(\boldsymbol{a}, \boldsymbol{A})$ is the Gaussian distribution with mean $\boldsymbol{a}$ and covariance $\boldsymbol{A}$. Strictly speaking, a simulation truth should be drawn from an initial distribution and not a set of means from a truth, but this is done such that the algorithm can be similarly initialized for any set of true

trajectories. Then, the initial PHD is of the form

$$v_0(\boldsymbol{x}_0) = \sum_{\ell=1}^{L_0} w_{\ell,0} p_g(\boldsymbol{x}_0\,;\, \boldsymbol{m}_{x,\ell,0}, \boldsymbol{P}_{xx,\ell,0})\,,$$

where $L_0 = 3$ and each $w_{\ell,0} = 1$.

Measurements are taken to be of the spherical coordinates range, inclination, and azimuth such that

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{v}_k = \begin{bmatrix} d_k & \theta_k & \phi_k \end{bmatrix}^T + \boldsymbol{v}_k\,,$$

where

$$\begin{aligned} d_k &= ||\boldsymbol{x}_k - \boldsymbol{x}_{s,k} + \boldsymbol{b}_{\text{loc}}|| + b_d \\ \theta_k &= \text{acos}(z_k/d_k) \\ \phi_k &= \text{atan2}(y_k, x_k)\,, \end{aligned}$$

$\boldsymbol{b}_{\text{loc}}$ is a bias in the location of the station (to account for orbit determination errors in the station state, for example), and b_d is a range bias. Measurements are collected every 10 seconds for half an hour as the targets drift away, and the sensor volume is taken to be a 10 km sphere around the station. The measurement noise $\boldsymbol{v}_k$ is generated according to a zero-mean Gaussian distribution with covariance $\boldsymbol{P}_{vv,k} = \text{diag}\{0.5^2, 0.573^2, 0.573^2\}$, with range in meters and angles in degrees. The generated observations can be seen in Figure 4.2.

The collection of parameters is taken to be

$$\boldsymbol{b} = \begin{bmatrix} C_d & \frac{A}{m} & J_2 & a & \omega & \psi & \boldsymbol{b}_{\text{loc}}^T & b_d \end{bmatrix}^T\,,$$

where C_d and $\frac{A}{m}$ correspond to the drag coefficient and area-to-mass ratio of the three identical satellites. Note that the satellites being identical is key to the GM consider PHD filter developed because, as formulated, the parameters cannot be unique to a specific target. For the station, $C_D \cdot \frac{A}{m} = 150$ kg/m^2 is used and is not treated as a parameter.

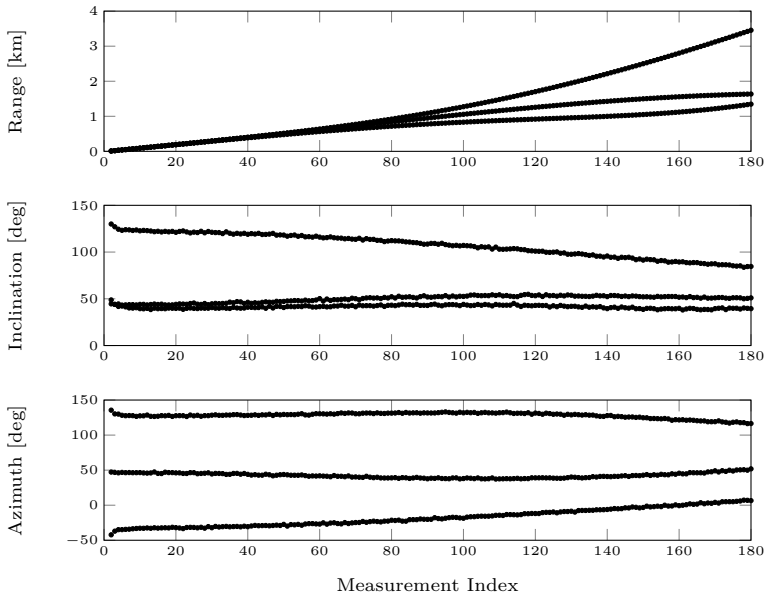


Figure 4.2. Measurement histories for the three targets.

The values of the elements of the (constant) parameter vector are taken to be

$$\boldsymbol{m}_{c,0} = \left[2.2 \;\; 0.01 \;\; 0.0011 \;\; \frac{\rho_0}{100} \;\; 2.5\sqrt{\frac{\mu}{r_0^3}} \;\; 0.9 \;\; \mathbf{0}_{1\times 3} \;\; 0.5\right]^T$$

with initial covariance

$$\boldsymbol{P}_{cc,0} = \mathrm{diag}\{(0.025)^2, (0.0005)^2, (10^{-8})^2, (10^{-17})^2, (10^{-4})^2, (0.005)^2, \mathbf{1}_{3\times 1}, (0.01)^2\}\,,$$

and true parameter values are then drawn according to

$$\boldsymbol{c}_{\mathrm{true}} \sim p_g(\boldsymbol{c}_{\mathrm{true}}\,;\, \boldsymbol{m}_{c,0}, \boldsymbol{P}_{cc,0})\,.$$

Two PHD filters are used to estimate the target states using the described measurement data: one filter *considers* the parameters and another *estimates* the parameters. Both filters utilize linearization to treat the system nonlinearities, and both filters utilize the same dynamical models as the truth. Note that, typically, when evaluating a filtering scheme, it is more realistic to utilize a lower fidelity model for the filters than the true

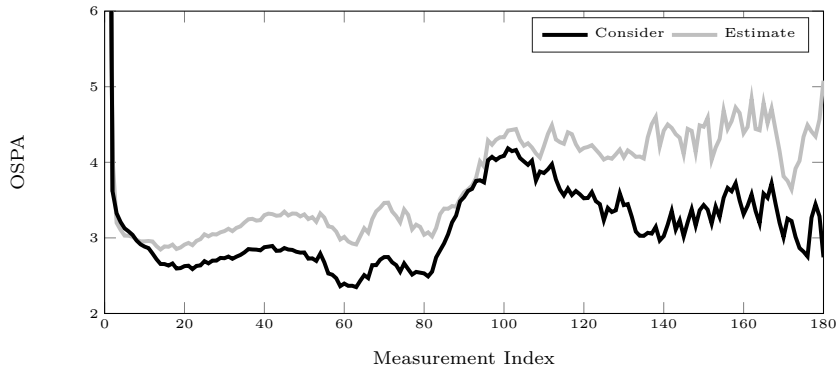


Figure 4.3. OSPA metric histories for the cases of considering and estimating the parameter vector.

model. However, as the performance of the GM PHD filter is well-presented in the literature, this simulation instead seeks to illustrate the impacts of considering versus estimating system parameters. Therefore, if the employed models match the true models, any difference in filter performance can be attributed to the difference in treatment of the stochastic parameters present. This is all to say that the fidelity of the two filters are the same such that performance differences can be attributed to the differences between considering and estimating the parameter vector.

In order to simplify the analysis of the results of the simulation, survival and detection are taken to be guaranteed, birth and spawning are neglected, and the mean clutter rate is taken to be zero (though clutter is observed to have little impact on the performance of the filter). The posterior GM PHD is pruned and merged at each time step. Components that are in 95% statistical agreement according to a χ^2 test are merged, and any components whose weight falls below 10^{-6} are removed. State extraction for the GM consider PHD filter, just like the standard GM PHD filter, is one of mode-finding, and in most cases is a very expensive problem. State estimates are taken to be the $\hat{N}$ highest weighted posterior GM components that satisfy $w_{\ell,k}^{+} > 0.5$ for both filters. Note that a convenient aspect of a GM PHD formulation is that the cardinality estimate $\hat{N}$ at any time is simply the sum of the GM PHD weights. Using the collected estimates of position, the OSPA metric $\bar{d}_p^{(c)}$ is computed at each iteration. The OSPA cutoff parameter c is taken to be 50 m, and selecting the order parameter $p = 1$ permits an "average per-target error" interpretation.

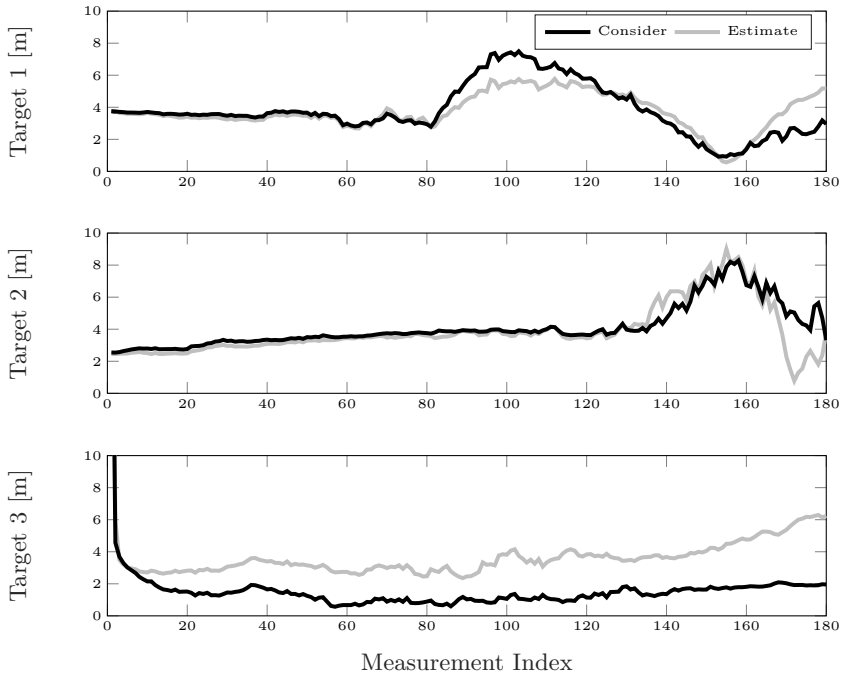


Figure 4.4. Target error histories for the cases of considering and estimating the parameter vectors.

The OSPA metric for the GM consider PHD filter (denoted by the black line labeled as "Consider") and the GM PHD filter that estimates the parameter values (denoted by the gray line labeled as "Estimate") can be see in Figure 4.3. Both filters perfectly estimate the target cardinality of three, but the consider-formulated PHD filter outperforms the filter that estimates the parameter values. Looking at the individual tracking errors presented in Figure 4.4, it is clear where the consider filter outperforms the filter that estimates the parameters. The standard GM PHD filter performs very similarly to the GM consider PHD filter for targets 1 and 2, occasionally outperforming the consider filter for a portion of the tracking period, but consistently produces larger estimation errors for target 3. At first this seems illogical, as it is natural to assume that estimating the parameter errors should outperform considering them with respect to error values. Additionally, one might expect that each target's estimation performance would be equally impacted by errors in parameter estimation (as the errors in targets 1 and 2 are very different than the errors in target 3). However, the final intensity estimate of the first element of the parameter $\boldsymbol{b}_{\text{loc}}$, which

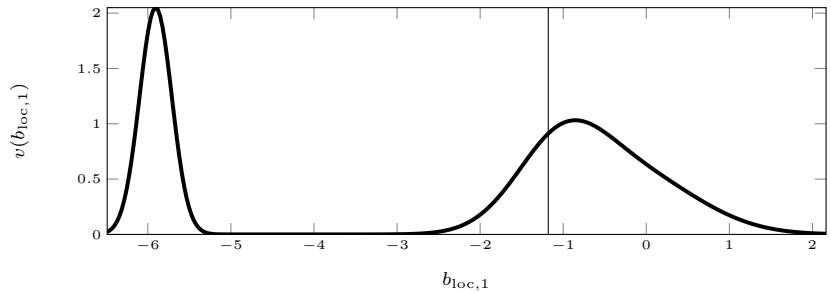


Figure 4.5. Final intensity estimate for $b_{\mathrm{loc},1}$ produced by the GM PHD filter that estimates the parameters with the true value denoted by the vertical line.

is shown in Figure 4.5, illustrates the cause of the diminished performance of the standard GM PHD filter. All of the true parameters are (unimodal) Gaussian random variables, and the multitarget RFS is modeled as Poisson; thus, the parameter PHDs are anticipated to be unimodal [87]. Here, however, it is seen that the standard GM PHD filter is producing multiple modes in its PHD estimate. This is because, due to the nonlinear measurement geometry, each target is contributing correlations in such a way that competing (different) parameter values are being deemed valid despite all targets sharing identical parameter values. Accordingly, the GM PHD filter for this example is using inappropriate (and incorrect) parameter estimates when tracking the targets, and this produces larger tracking errors. This is a theoretical shortcoming that is not experienced by the GM consider PHD filter. It is a matter of problem geometry that the tracking performance of each target is impacted differently by this incorrect parameter estimation because, in this case, the nonlinear measurements of target 3 are more sensitive to this location bias than the other two targets due to its position with respect to the sensor.

4.3. APPLICATIONS TO NAVIGATION

In many navigation applications, it can be beneficial, or even necessary, to utilize sensor types such as cameras or lidars to gather information pertaining to features within a vehicle's environment and, ultimately, produce improved estimates of the vehicle's navigation state. This technique, herein referred to as terrain aiding, has been a focus of

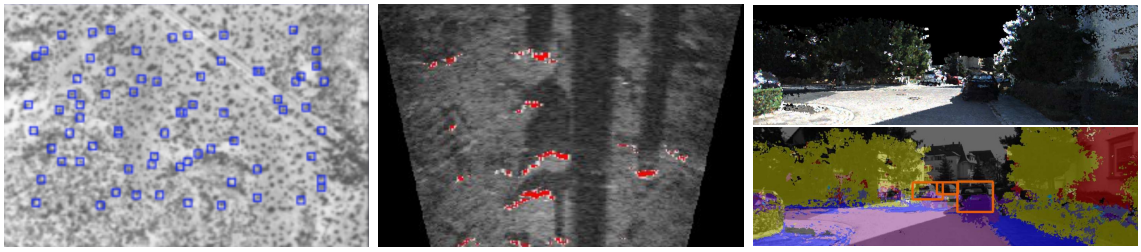


Figure 4.6. Examples of terrain aiding in a navigation context in applications of (left) a terrestrial sounding rocket test [106], (center) underwater harbor surveillance [107], and (right) autonomous passenger vehicles [108].

research for decades [105] and has facilitated useful navigation capabilities in terrestrial sounding rocket tests [106], underwater harbor surveillance [107], and autonomous passenger vehicles [108], just to name a few (each depicted in Figure 4.6). In many applications, these techniques are a powerful augmentation of existing, data-rich navigation sources, like GPS, and produce beneficial improvements to standard navigation methodologies. In non-terrestrial applications, such as planetary descent and landing where critical information sources such as GPS are unavailable, terrain aiding becomes *necessary* to achieve pinpoint landing [106]. Even in terrestrial applications, certain vehicles, such as guided missiles or drones, may similarly be denied from rich information sources like GPS.

Terrain aiding can come in a number of varieties, but the concept is the same: collect information regarding the vehicle's environment and somehow use the information to improve the navigation performance of an onboard filter. Some methods utilize optical flow to use visual sensors as a "rate sensor" in order to stave off growth of uncertainties during times of propagation [109]. While this is useful in that context, it does not permit reduction of uncertainty, and, rather, it allows reduction in the rate of growth of uncertainties. Other methods utilize a known, fixed reference map and correlate it with collected camera images/lidar data/etc. and use projective geometry to meaningfully update the relative position information of the vehicle. This is an incredibly powerful measurement type, but it relies on a number of limiting and hazardous assumptions that can devastatingly degrade the vehicle's navigation performance. In particular, these assumptions usually result in incorrectly modeled measurement uncertainties, poorly described detection processes, overly

optimistic reliance upon a reference map, loss of a vehicle's exploratory authority (i.e. a vehicle's ability to autonomously explore) due to this map reliance, and more. An example of this type of interpretation, and a proposed solution, is the subject of Section 5.

Instead, one can define a dynamic map and correlate that map with the vehicle state, resulting in the well-known simultaneous localization and mapping (SLAM) problem. That is, define a map that contains a varying number of (possibly moving) uncertain features, compare any available *a priori* information about the map and the vehicle state with collected observations, and update both the map and the vehicle state using accumulated correlations. The reason for doing this is that, due to the potentially varying number of uncertain features within the map, an *a priori* reference map can be employed but is not required. Furthermore, using FISST, the measurement and detection process can be more fully and rigorously modeled, in addition to permitting discovery of new map features (i.e. target birth). Of course, there is no free lunch, and this approach inherits a new problem: the data association/multitarget tracking problem. How does one say that a collected measurement should be applied to one target over another? Fortunately, as already discussed in this dissertation, a number of useful approximate filtering tools exist to accomplish multitarget tracking, and this section will leverage such results directly.

As an example, consider the lander traversing an uncertain terrain shown in Figure4.7. This lander takes noisy optical measurements of the terrain within a limited field of view (illustrated as the black traces on the surface). The "Reality" panel illustrates what the "real world" looks like: the lander travels from time t_1 to t_2 along the dotted path, taking observations and making decisions based on those observations. The range of colors in this panel only illustrates the variance in height of the terrain below. The terrain contains features that can be interpreted as hazards or safe zones, such as peaks, valleys, and plateaus. By observing these features and estimating their locations, the vehicle is able to refine estimates of the map and its state estimate simultaneously. The panels "Framework (t_1)" and "Framework (t_2)" illustrate what the flight computer "sees" as the lander traverses the terrain. During this transit, features are detected (or misdetected) by onboard sensors, and these measurements are to be processed by the filter. The true feature locations are denoted by $\times$'s, and the confidence in their estimates are represented by the map below, where

bright yellow, concentrated regions on the surface represent high confidence and darker colors represent lower confidence. Through the refinement of the feature locations, i.e. as the onboard navigation filter receives more data, the navigation solution of the vehicle itself is improved.

This section describes how SLAM tools derived using FISST, such as the methods described in [99, 102, 110], can be used for terrain aiding in navigation. This is far from the first look into the use of FISST for SLAM, but the following investigates a particularly attractive approximation that enables practical navigation capabilities. The sensor that is observing the terrain around the vehicle will commonly be referred to as a "terrain camera" for convenience, but there is no need that it be a camera. Indeed, any sensor can be used, so long as detections correspond to map features such that a given feature generates at most one measurement in a given collection.

4.3.1. A Useful Approximation. The most widely published approximation to FISST-based SLAM is that of using sequential Monte Carlo (SMC) techniques with Rao-Blackwellization, where an ensemble of trajectory samples are drawn and a filter estimates a trajectory-conditioned map for each sample [102]. To mitigate some of the intense computational cost required by SMC-based approaches, a different approximation is adopted that is akin to the "brute force" approximation of Mullane et al. in [110]. Interestingly, it is mentioned in [110] that the "brute force" approximation was abandoned in favor of the SMC methods due to the computational savings, but the following aims to demonstrate that, at least for the problems considered here, this has precisely the opposite effect. It is the author's belief that, while the SMC methods with Rao-Blackwellization are more mathematically elegant, the approximation adopted herein is much more practical, and, indeed, is found to be more useful in the numerical navigation studies of Section 4.3.5.

Given N_k terrain features and a set of D_k feature observations, their RFSs are written as unordered sets as

$$\boldsymbol{M}_k = \{\boldsymbol{\zeta}_{k,1}, \boldsymbol{\zeta}_{k,2}, \ldots, \boldsymbol{\zeta}_{k,N_k}\} \qquad \text{and} \qquad \boldsymbol{Z}_k = \{\boldsymbol{z}_{k,1}, \boldsymbol{z}_{k,2}, \ldots, \boldsymbol{z}_{k,D_k}\},$$

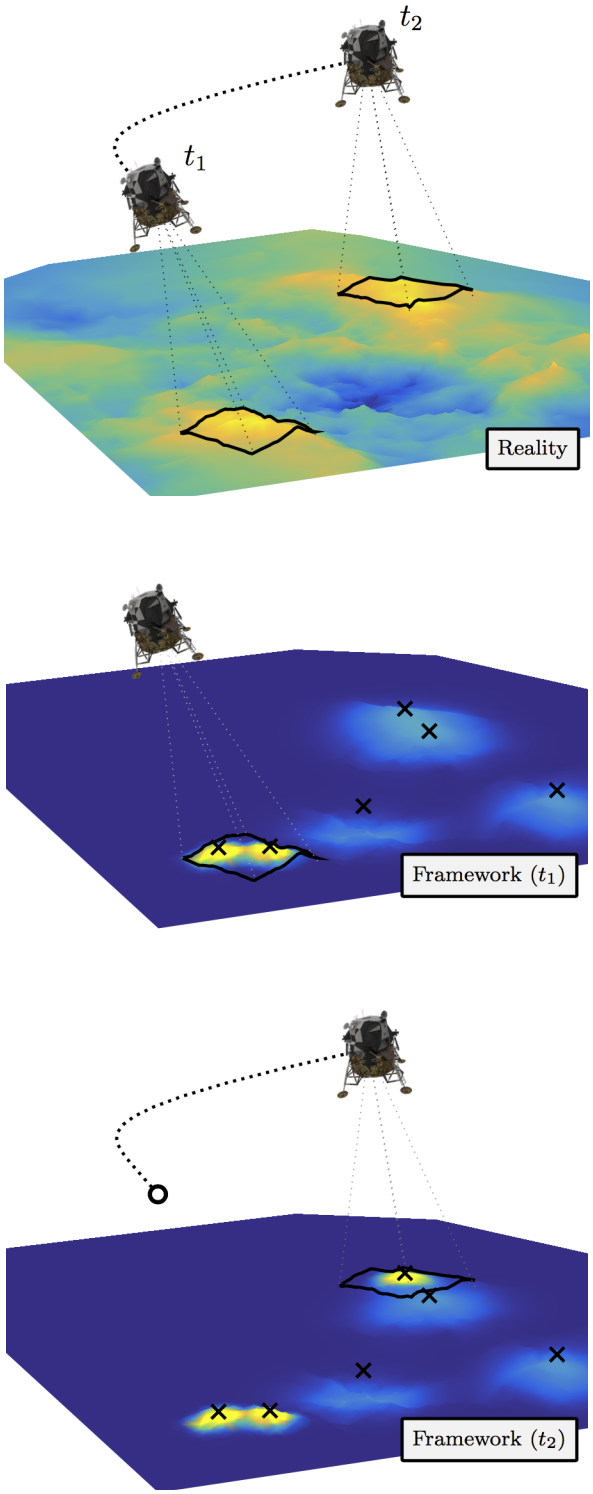


Figure 4.7. Illustration of a lander traversing an uncertain terrain.

respectively, and are called the map state and the map-based observation, whose elements belong to feature space $\mathbb{M}$ and observation space $\mathbb{Z}$, respectively. Each $\boldsymbol{\zeta}_{k,i} \in \boldsymbol{M}_k$ is a vector describing a feature within the map. The map-based observation contains both map-originated and clutter-originated sensor returns from the terrain camera, but it provides no information on the specific source of each sensor return.

Let $\mathbf{x}_k \in \mathbb{R}^{n_\mathbf{x}}$ denote the vehicle state; that is, $\mathbf{x}_k$ is a vector containing appropriate parameterization of the vehicle, such as position, velocity, attitude, and any estimated parameters.[10] Furthermore, let the *vehicle-map RFS* be defined as

$$\boldsymbol{X}_k = \{\boldsymbol{x}_{k,1}, \boldsymbol{x}_{k,2}, \ldots, \boldsymbol{x}_{k,N_k}\},$$

where each *vehicle-map state* $\boldsymbol{x}_{k,i} \in \boldsymbol{X}_k$ is a concatenation of the vehicle state and a map state of the form

$$\boldsymbol{x}_{k,i} = \begin{bmatrix} \mathbf{x}_k \\ \boldsymbol{\zeta}_{k,i} \end{bmatrix} \tag{4.11}$$

and $\boldsymbol{X}_k \subset \mathbb{X} = \mathbb{R}^{n_\mathbf{x}} \times \mathbb{M}$. All map states $\boldsymbol{\zeta}_{k,i}$ are tracked in the body-fixed frame of the terrain surface to which they belong. Moving forward, the subscript "i" will be dropped from $\boldsymbol{x}_{k,i}$ and $\boldsymbol{\zeta}_{k,i}$ in cases where there is no risk of confusion. Note that this *ad hoc* definition is an abuse of theory, as the vehicle state has been "smeared" across the distinct map states. Equation (4.11) is the principal component of the approximation employed here and is the key to producing a practical navigation implementation.

This approximation is illustrated in Figure 4.8. The left panel depicts a mobile sensor collecting observations of three map features. In the top of the right panel, the vehicle state is represented as a vector, and the state of the map features are contained within a set. The bottom right panel shows the key to the approximation, where the vehicle state is concatenated with each of the feature map states and the resulting vectors are treated as elements of a set.

[10] Note that consider parameters are also permissible under this formulation, and one simply needs to identify which parameters in $\mathbf{x}_k$ should be estimated and which should be considered.

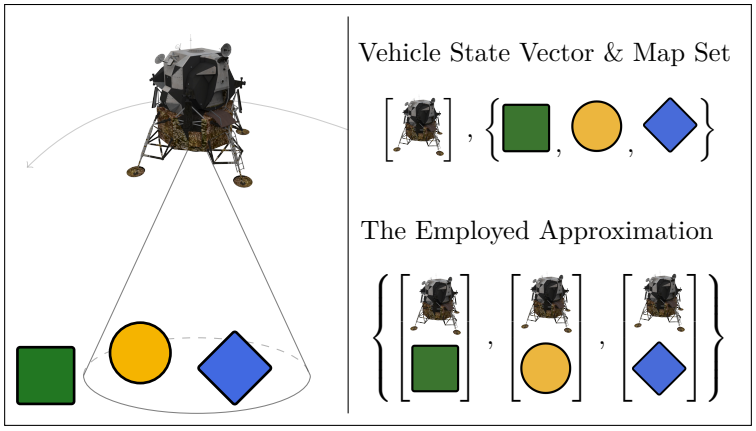


Figure 4.8. Illustration of the approximate formulation.

Given this formulation, one can form the multitarget Bayes filter directly and approximate it however they like. This means that, given the approximation of Eq. (4.11), a standard PHD, CPHD, multitarget multi-Bernoulli, LMB, δ-GLMB, etc. filter can be designed to estimate $\boldsymbol{x}_k \in \boldsymbol{X}_k$. Some modifications are required, but this illustrates one powerful component of the employed approximation: one does not need to derive new filters to conduct navigation. Instead, one can make principled modifications of existing filters, and such modifications will be described in the following sections.

The computational advantage of the approximation in Eq. (4.11) over the SMC approach for these problems is that, as more and more data are processed, the model complexity required to characterize the multitarget density should be expected to reduce. This is particularly important when one considers that nearly every vehicle performs high frequency time updates using IMU data. While not particularly costly per time step, these time updates amount to a substantial portion of a navigation filter's computing budget due to the sheer amount of data required to maintain accurate state estimates. So, rather than relying on a large ensemble of samples for the entire length of the mission, complexity reduction techniques, such as GM pruning and merging [69], can be used to greatly reduce the number of model components and, accordingly, the number of required IMU-based time updates. The difference between the SMC method and the presented approximate method is graphically represented in Figures 4.9 and 4.10, respectively. With the SMC method, a

map representation is maintained for each of the N samples (e.g. operating N PHD filters at the same time), and this number of samples stays the same through time, as seen in Figure 4.9. The approximation employed by this work, as seen in Figure 4.10, permits samples that are deemed statistically similar to be merged, ultimately producing a less complex parameterization and therefore reducing computational complexity as more data are collected.

It is important to note that there is nothing inherently flawed about the SMC implementations and that, in fact, they are much more mathematically elegant than the approximate method described here. However, these methods were primarily developed with ground robotics in mind, such as self-driving cars, where the system is mostly influenced by well-known control inputs. In contrast, the types of problems considered here are much more beholden to the highly complex and uncertain dynamics of the system they belong to, such as orbital mechanics and aerodynamic drag, and, therefore, a much more dense SMC sampling is required to characterize the motion. This translates to an aggressive growth in complexity that makes it infeasible for these types of navigation applications. Instead, the approximation, in very loose terms, more efficiently describes the vehicle's state approximation by using GM components as "fattened" particles. Furthermore, it is reiterated that the approximate method permits complexity reduction through time, providing additional feasibility for onboard processing applications.

As the number of available filtering tools in this arena is immense, the simulations of Section 4.3.5 study the PHD and δ-GLMB filters as "bookends" to illustrate the capabilities of "very simple" and "very complex" implementations, respectively. This is not to imply these are the only available tools or that these truly are the simplest or most complex of the available tools, but to instead illustrate that, given this formulation, any FISST-based filter can be used. These filters are not explicitly presented here as they are not innovations of this work and there is ample available literature. One must modify the related expressions to conform with the approximation of Eq. (4.11), but most of the mechanization of these filters remains the same. Accordingly, these filters will be referenced repeatedly, but a user versed in FISST-based filtering will have no trouble adapting for a given filter. Of

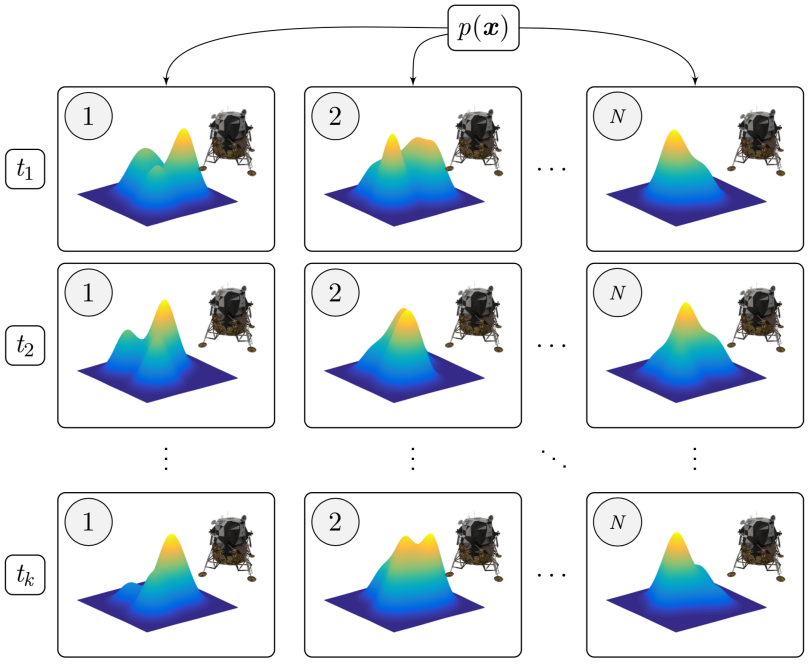


Figure 4.9. Schematic representation of the SMC methods for terrain aiding with FISST.

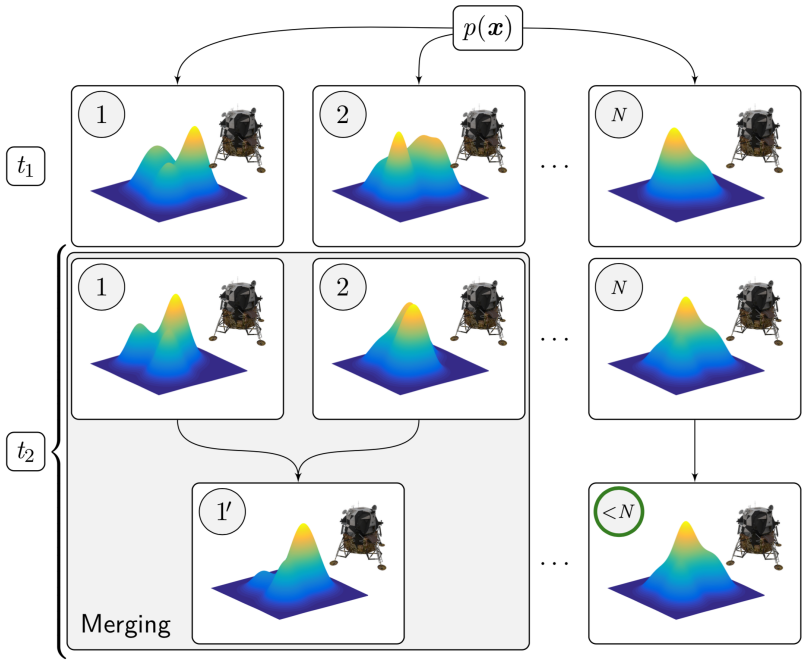


Figure 4.10. Schematic representation of the approximate method for terrain aiding with FISST, where it is emphasized that the model complexity is permitted to reduce through time.

course, the square-root and consider filtering techniques discussed in Section 2, Section 3, and Section 4.2 could instead be applied if a multitarget square-root and/or consider filter is desired.

Of principal importance are the required modifications to *initialization* and *state estimation*. Building appropriate mathematical expressions for the predictor, corrector, and maintenance (that is, the mechanisms that prevent exponential growth of complexity of the methods) steps should follow directly from this discussion and the relevant references, particularly as described in [69, 72, 73].

4.3.2. Initialization. Regardless of the filter selected to process the terrain camera data, initialization can be summarized as being comprised of three stages:

- trajectory sampling,
- map initialization, and
- GM construction.

Trajectory Sampling. Inspired by the SMC methods, an initial collection of N trajectory particles is drawn according to an initial vehicle density as

$$\mathbf{x}_0^{(j)} \sim p(\mathbf{x}_0)\,, \qquad j \in \{1, \ldots, N\}\,.$$

The choice of initial distribution is up to the operator, but it is common to use a Gaussian distribution with some mean and covariance that are the product of some initialization process, such as ground-based trajectory determination and onboard attitude calibration before mission initiation. These samples are used to develop a collection of candidate vehicle trajectories, much like the SMC methods. In contrast, however, the approximation of Eq. (4.11) permits the filter cost to reduce as more measurements are processed by the algorithm, as the employed model simplification methods (here termed "maintenance") permit fewer and fewer competing hypotheses to be managed by a filter. The result is a much more computationally feasible algorithm for real-time implementation, particularly due to the reduction in the number of required IMU-based time update steps when IMUs are employed.

Map Initialization. Each feature $\boldsymbol{\zeta} \in \boldsymbol{M}_k$ that is known *a priori* is taken to be described by a GM as

$$p_i(\boldsymbol{\zeta}) = \sum_{\ell=1}^{L^{(i)}} w_{\ell,0}^{(i)} p_g(\boldsymbol{\zeta}; \boldsymbol{m}_{\ell,0}^{(i)}, \boldsymbol{P}_{\ell,0}^{(i)}), \qquad i \in \{1, \ldots, M_0\}$$

where M_0 is the number of map features in the initialized map. That is, for each feature $\boldsymbol{\zeta}$ in the *a priori* map, such as from information collected on previous reconnaissance, loaded surface data files, etc., there is a corresponding $L^{(i)}$-component GM describing its state density. In the context of this study, this means that $p_i(\boldsymbol{\zeta})$ is the pdf of a map features position in the fixed frame, and if the loaded reference map has M_0 features in it, there will be M_0 such GMs. In a practical application, it is likely that this collection of GMs will reduce to a collection of Gaussians (i.e. each feature is described by a single Gaussian), but the GM representation is retained to permit features with complex geometries, such as a rocky ridge or a composite crater consisting of multiple circular regions.

GM Construction. This stage of initialization is the only one of the three that depends on the type of FISST-based filter employed. However, the concept is the same and extends conceptually to any filter design. Here, the concept of this stage is outlined generally, and then, specifics for PHD and δ-GLMB filters are outlined to aid in a reader's implementation of these methods.

Regardless of the employed filter or parameterization, the idea is to create a copy of all *a priori* map information for each of the N vehicle state samples drawn in the first stage of initialization. So, given M_0 densities describing the *a priori* map, N copies of these densities are to be made, each associated to one of the N vehicle state samples. This is done according to Eq. (4.11) by concatenating each of the components of the i^{th} map feature densities, for $i \in \{1, \ldots, M_0\}$ with the j^{th} vehicle state sample, for $j \in \{1, \ldots, N\}$. This is where the previously described "smearing" of the vehicle state across the map state occurs, and the result is the ability to naturally estimate the correlations between the vehicle state and the map features.

For the PHD filter, this initialization is quite handily performed. The initial PHD of the features is simply the sum of the densities of all the features, i.e.

$$v(\boldsymbol{\zeta}) = \sum_{i=1}^{M_0} p_i(\boldsymbol{\zeta})\,.$$

Then, the PHD used to initialize the filter is obtained by creating N copies of $v(\boldsymbol{\zeta})$ and, for each $j \in \{1, \ldots, N\}$, concatenating the mean and covariance of each sample $\mathbf{x}_k^{(j)}$ in the manner of Eq. (4.11). The result is a PHD defined on the state space of $\boldsymbol{x}_k$, and subsequent predictor, corrector, and maintenance stages can be performed precisely as described in [69].

For numerical implementation, initializing a δ-GLMB filter requires four things: a collection of track densities, a set of hypothesis-track labels, hypothesis weights, and a cardinality distribution. The collection of track densities is obtained in much the same way as the PHD filter: for each feature, its GM pdf is copied N times, each of the GM components in each copy is concatenated with the mean and covariance associated with $\mathbf{x}_k^{(j)}$ in the manner of Eq. (4.11), and the result is associated with a track label ℓ. Each initial track label $\ell \in \{1, \ldots, M_0\}$ is taken to be distinct, and the collection of all ℓ comprises the complete set of initial hypothesis-track labels. This is to imply that the δ-GLMB filter is initialized with a *single* hypothesis of M_0 tracks in this way, and accordingly, its associated hypothesis weight is set to one. Finally, the cardinality distribution is set such that the probability of $|\boldsymbol{M}_k| = M_0$ at $k = 0$ is equal to one.

4.3.3. Sequential Filtering. As mentioned before, the filtering stage follows the standard filtering recursion of the selected multitarget filter. No modification of the filtering equations needs to be made to conduct predictor/corrector stages and process data. However, it is important to note that the approximation presented here permits not only terrain-related data to be processed, but vehicle-specific data, such as GPS, barometers, etc., can be processed within the architecture as well. If vehicle-specific data are obtained, one can simply treat it as a singleton observation set, i.e. $\boldsymbol{Z}_k = \{\boldsymbol{z}_k\}$, where $\boldsymbol{z}_k$ is the vector

of collected vehicle-specific data with known measurement noise, and process it using a slightly modified GM filter update. The result is a "hybrid filter" that uses a multitarget filter for the map-related data and a modified GM filter filter for the vehicle-specific data.

For a PHD filter implementation, the vehicle-specific data can be processed using GM filter equations, such as the GM consider filter, for example, as defined in Section 3.3.3 using Eqs. (3.53) on pp. 123. By computing the Jacobians $\boldsymbol{H}_{x,k}$, $\boldsymbol{H}_{c,k}$, and $\boldsymbol{H}_{v,k}$ corresponding to the process that generated the vehicle-specific data, Eqs. (3.53) on pp. 123 are applied to the components of the GM intensity function. The only exception/modification to these equations is the weight update, Eq. (3.53a), which is instead taken to be

$$w_{\ell,k}^{+} = \hat{N}_{k}^{-} \cdot \frac{w_{\ell,k}^{-} q_{\ell,k}(\boldsymbol{z}_k)}{\sum_{j=1}^{L} w_{j,k}^{-} q_{j,k}(\boldsymbol{z}_k)},$$

where, as usual, $\hat{N}_{k}^{-}$ is the sum of the *a priori* GM weights. The reason for this modification is that, unlike the GM pdfs considered in Section 3.3.3, the GM now represents a multitarget PHD and, therefore, does not possess weights that sum to one. Therefore, since the vehicle-specific data should not necessarily impact the cardinality estimate of the map, the weights are normalized as usual and then re-scaled with the *a priori* cardinality estimate.

For a δ-GLMB implementation, one can directly apply Eqs. (3.53) to each of the tracks of the filter without modification because each track is represented using a proper pdf, given that Jacobians are appropriately defined.

As an illustrative example, take the vehicle state to be position and velocity of the form $\mathbf{x}_k = [x \;\; \dot{x}]^T$ and the map state to be scalar of the form $\boldsymbol{\zeta} = \zeta$ such that the vehicle-map state is given as $\boldsymbol{x}_k = [x \;\; \dot{x} \;\; \zeta]^T$. Let the map-related data be generated according to the function $h^{(\text{map})}(\zeta) = \zeta$, i.e. the feature position is measured directly, and the vehicle-specific data is generated according to the function $h^{(\text{veh})}(\mathbf{x}_k) = x$, i.e. the vehicle position is measured directly. Then, if map-related data are processed, the measurement mapping matrix in the standard filtering equations is

$$\boldsymbol{H}_{x,k}^{(\text{map})} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix},$$

and if vehicle-specific data are processed, the measurement mapping matrix is

$$\boldsymbol{H}_{x,k}^{(\text{veh})} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}.$$

This means that both terrain-related and vehicle-specific data can be processed by this scheme using a "hybrid filter" construction. One could similarly process vehicle-specific data if using an SMC-based method, but this would require a particle filter-style of measurement update and, as discussed previously, particle-driven methods are not generally amenable to navigation applications.

4.3.4. Vehicle and Map Estimation. Regardless of the filter selected to process the terrain camera data, vehicle and map state estimation can be summarized as comprising of two stages:

- vehicle-map state estimation and
- marginalization.

Vehicle-Map State Estimation. Despite the modifications to the PHD procedure to produce a SLAM recursion, some of the salient elements of state estimation remain unchanged. First, the estimated cardinality of $\boldsymbol{X}_k$ is still given as the sum of the weights of the GM intensity; that is, $\hat{N}_k = \sum_{\ell=1}^{L_k} w_{\ell,k}$, where L_k is the number of GM components and $w_{\ell,k}$ is the ℓ^{th} component's weight. Additionally, multitarget state estimation remains conceptually unchanged from a standard PHD filter: local maxima of the PHD approximation correspond to state estimates. In the map estimation employed here, only portions of the GM components that correspond to the map features are considered. While conceptually very clear, performing mode-finding for an arbitrary GM is a difficult and expensive numerical procedure in practice. A common approach is to accept the $\hat{N}$ highest weighted GM components that are above some threshold (0.5, for example) as an efficient state estimation strategy [69]. The result is a collection of vectors defined on the vehicle-map state space, and stripping the portions of these vectors corresponding to the vehicle state $\mathbf{x}_k$ produces the PHD filter's map estimate.

In contrast, the δ-GLMB filter's state estimates are directly obtained from the δ-GLMB parameter set. Map estimates are acquired by finding the most likely cardinality from the cardinality distribution of the δ-GLMB RFS and then accepting track estimates from the highest weighted hypothesis with that cardinality. Again, as with the PHD map estimation, only components of the concatenated vehicle-map track density that correspond to the map features are considered as map estimates. For map estimates from each individual track's GM, the conditional mean of each GM is accepted as that track's state estimate, computed using the method of moments. One could instead accept the maximum likelihood estimate of the GM, but, again, this is a costly mode-finding problem and is avoided here.

Marginalization. Obtaining an estimate of the vehicle trajectory for the PHD filter follows by taking the GM intensity in the vehicle-map state $\boldsymbol{x}_k$ and marginalizing out the map terms, $\boldsymbol{\zeta}_k$. The result is a weighted collection of Gaussians (a GM) in the vehicle state $\mathbf{x}_k$ only; in fact, it is a proper density of the vehicle state! Then, any conventional GM estimation technique can be employed, such as mode-finding or the method of moments. Intuition indicates that the true vehicle density should be unimodal (or at least nearly-so), such that it would be well represented by a mean and covariance pair, and, accordingly, this work accepts the conditional mean and covariance produced by the method of moments for simplicity.

For vehicle state estimation with the δ-GLMB approach, all of the track GMs corresponding to each hypothesis are linearly combined, each weighted by the corresponding hypothesis weight, and the map features are marginalized out of the resulting GM. Effectively, the result is a combination of all the vehicle-related portions of each hypotheses' GM track densities weighted by that hypothesis' probability. This weighted combination of GMs produces a GM that can then be used for vehicle state estimation. Instead, one could choose to utilize the single highest-weight hypothesis' GM found the vehicle-map state estimation, especially in a case where computational resources are more limited. This work utilizes the conditional mean and covariance produced by the method of moments for vehicle state estimation under the belief that the true vehicle state density should be relatively well-approximated by a single Gaussian.

Alternatively, for any general FISST-based filtering method, a GM representation of the vehicle state can be maintained for state estimation, rather than that of a Gaussian, but this depends on the specific flight processing capabilities of the mission.

Summary of Approximate Method. First, select a FISST-based filter and employ the approximation of Eq. (4.11). Initialize the filter via sampling and concatenation as described in Section 4.3.2, and perform predictor/corrector stages as defined by the selected filter using the concatenated states and appropriate mapping matrices as described in Section 4.3.3. Take advantage of GM pruning and merging to substantially reduce the computational burden of the required IMU-based time update steps and measurement updates. When estimation is required, follow the steps of Section 4.3.4 to obtain map and vehicle state estimates.

4.3.5. Simulation Studies. To investigate the performance of the proposed navigation scheme, consider again the ballistic trajectory example described in Section 3.2.6 and depicted in Figures 3.3a and 3.3b on pp. 103. Now, however, rather than processing data from a ground-based radar, the ballistic vehicle is equipped with a camera to perform terrain-aided navigation onboard. While this scenario was used previously to illustrate ground-based tracking capabilities, this dynamical system serves as an excellent candidate to test navigation problems. This is because many challenging navigation problems, such as vehicle re-entry, landing, and intercept, possess common traits, such as typically having (a) large initial state uncertainties, (b) large dynamical modeling errors due to ballistic coefficient uncertainties, atmospheric uncertainties, and process noise effects, (c) short windows of time within which to collect data, and (d) denied GPS capabilities, just to name a few. Indeed, this scenario contains each of these ingredients and, thus, serves as an excellent testbed for new navigation techniques. The only missing component to complete the analogy to practical navigation is the use of IMU-based time updates, but these are utilized extensively in the navigation application in Section 5 to further demonstrate the capabilities of the approximation.

Using an onboard camera, the vehicle collects images of features in the terrain below as it descends, and these measurements are assumed to be in the form of pixel coordinates of features within an image. Since this is a two-dimensional/planar example, the term “image”

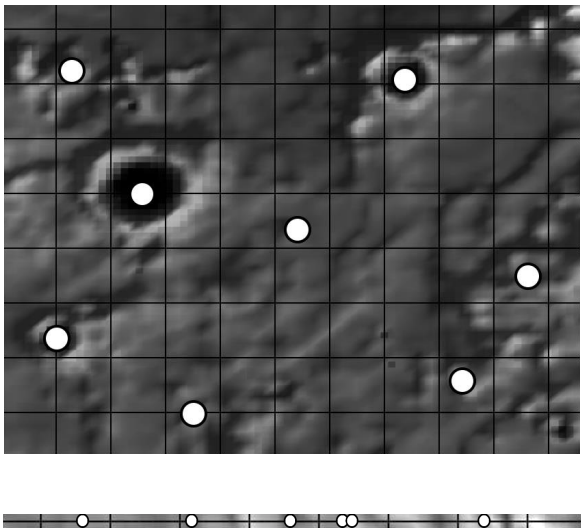


Figure 4.11. Comparison between standard images (top) and the one-dimensional images employed in the present example (bottom), where white circles correspond to pixel locations of map features.

is used as a representative facsimile of images collected in a three-dimensional example (i.e. standard images). This is illustrated in Figure 4.11. In standard images collected with digital cameras, an electronic sensor within a camera's lens assembly produces a two-dimensional image such that feature coordinates are in the form of (i, j) pixel pairs. Here, an image is a "slice" of a standard image, such that images are one-dimensional and feature coordinates are in the form of a single pixel, i, per feature.

A simulated surface and features are generated below the ballistic trajectory depicted previously in Figure 3.3a. To do this, a surface is arbitrarily defined according to the function

$$y(x) = 150\sin(5\times10^{-5}x)\cos^3(5\times10^{-5}x) - 900\sin(1.25\times10^{-4}x)\,,$$

and 20 random samples along this surface are taken as features comprising the map. This surface and resulting features are depicted in Figure 4.12a. The camera onboard the vehicle is designed such that it collects pixel coordinates of features within 20° of the camera's pointing direction, and this pointing direction is taken to be 45° below the x-axis direction. As can be seen in Figure 4.12b, this means the vehicle has the camera pointed "behind

it" in the manner shown by the dotted lines, such that any feature within the dotted lines is a candidate for detection. If a feature is within the field of view (FOV), it is detected according to some (state-independent) probability of detection, $p_{D,k}$.

Recall that the position of the vehicle is given by coordinates x and y, and let the position of a given feature be given by the coordinates x_ζ and y_ζ. Then, a feature's image coordinate, in terms of pixels, is given as

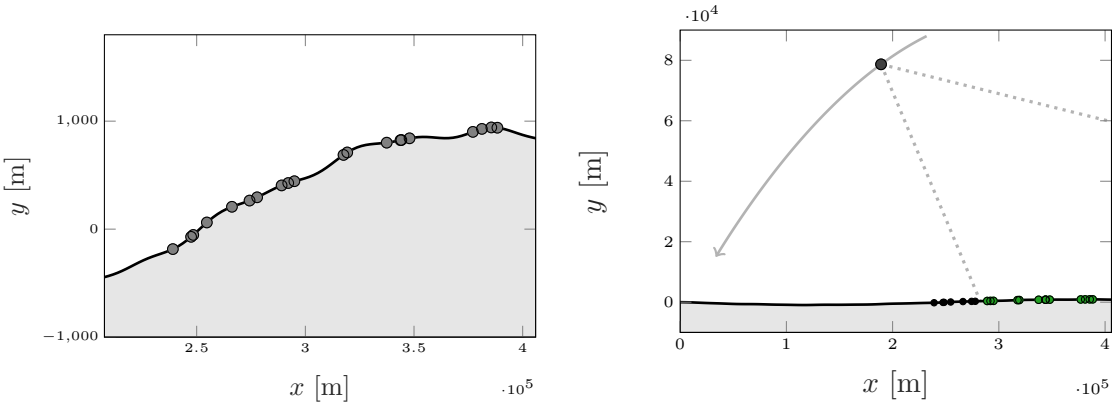
$$i = f_c \cdot \frac{x - x_\zeta}{y - y_\zeta} \,,$$

where f_c is the "focal length" of the camera, and in all simulations to follow, $f_c = 1080$ pixels. Note the similarity between this and a standard pinhole camera model. Indeed, this is a pinhole camera model for one-dimensional images, and, therefore, serves as a representative analog to real-world problems. Collected image coordinates are taken to be corrupted by a constant bias and zero-mean white noise, such that the collected measurements are of the form

$$z_k = f_c \cdot \frac{x_k - x_\zeta}{y_k - y_\zeta} + b_{\text{cam}} + v_k \,,$$

where the bias and noise standard deviations are given by $\sigma_{b,k}$ and $\sigma_{v,k}$, respectively.

This scenario is deliberately designed to be a challenging navigation problem, serving as a stress test the proposed methods. First, note that, as shown in Figure 4.13, about 60 seconds after the initialization of the trajectory, the air density becomes appreciable and exponentially increases. Soon afterward, this produces an exponential increase in the aerodynamic drag experienced by the vehicle, about 70 seconds into the trajectory, and since the ballistic coefficient is uncertain, so are the drag accelerations. These uncertain accelerations result in highly uncertain vehicle motion and, if unabated by processing measurements, produce vast state uncertainties of approximately $17,000$ m and 650 m/s in position and velocity, respectively. On the other hand, if measurements are processed



(a) Terrain and map features

(b) Sensing scheme

Figure 4.12. Illustration of the 20 simulated terrain and map features (left, y-axis enhanced to show detail) and the observational scheme for the ballistic trajectory (right) where currently observed features (within the dotted lines) are highlighted in green.

during this high-acceleration phase, it turns out that, as demonstrated in the simulation of Section 3.2.6, standard filters become highly unstable, motivating the use of the square-root consider methods employed here.

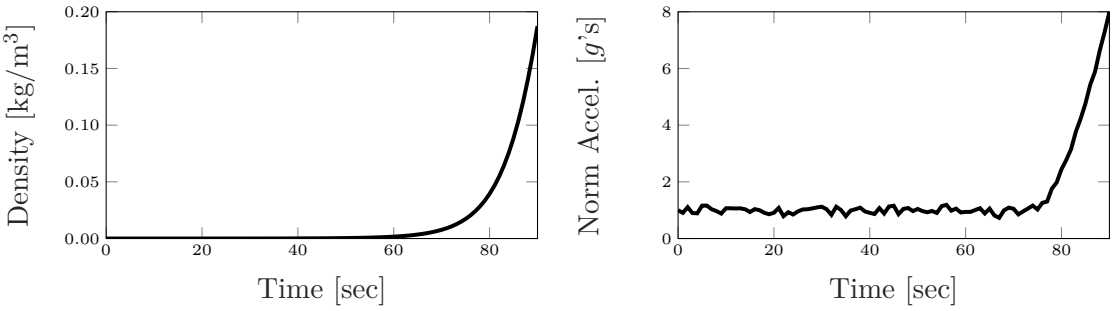
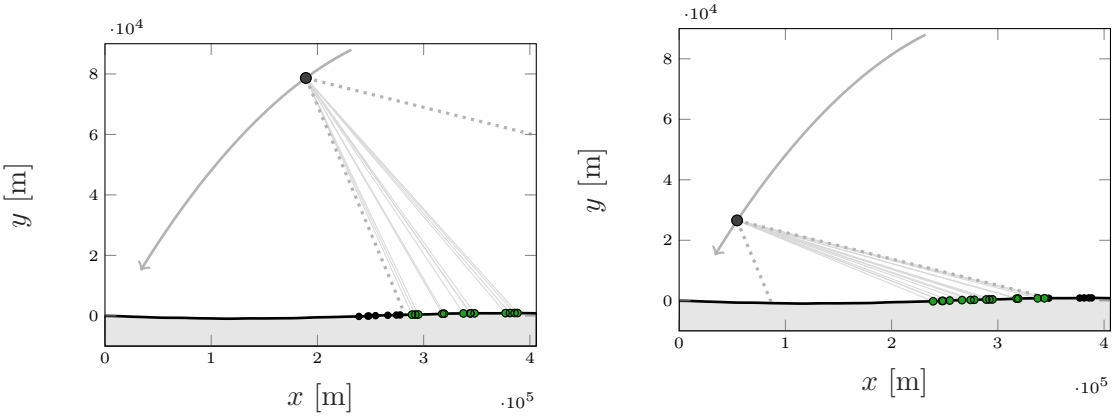


Figure 4.13. Density and acceleration during the trajectory.

Second, the map is designed such that the measurement geometry poses challenges to the multitarget filter. As depicted in Figure 4.14, the lines-of-sight from the vehicle to the map features, and, accordingly, the corresponding pixel coordinates, begin to coalesce as it descends. Toward the end of the trajectory, each of the map features, as seen by the vehicle, are nearly on top of each other, resulting in pixel coordinates that are very near together. Given these observations and nothing more, the filter is not provided with enough information to meaningfully distinguish which measurement should be applied to which fea-



(a) Scan at $t = 20$ seconds.

(b) Scan at $t = 80$ seconds.

Figure 4.14. Depiction of the coalescence of lines-of-sight, thus inducing confusion between measurement assignments.

ture in its *a priori* map. Given that these types of events are rigorously modeled within the FISST framework, this is not a filter stability problem, but, instead, is a classic stress event for multitarget filters. In essence, the correct association becomes almost equally ambiguous among the remaining association hypotheses (i.e. it becomes difficult to discern which are the most likely), and in the event that all associations are almost all weighted equally, measurements become nearly meaningless. For example, while the PHD filter obviates the need for explicit data association, it directly weights data on the state space according to the measurement likelihood function. If all of the targets are "stacked" within the measurement space, each target will receive nearly the same weight update despite the fact that, say, Target A's measurement is being erroneously applied to Target B. Alternatively, in these ambiguous situations, something like the δ-GLMB filter will need to maintain increasingly more association hypotheses as subsequent data are collected, either (a) requiring more computational resources or (b) substantially degrading tracking performance by discarding valid hypotheses.

As in Section 3.2.6, the vehicle state is defined as comprising of the position and velocity of the vehicle, but now the ballistic coefficient (β) and terrain camera bias (b_{cam}) are treated as consider parameters. Since numerical stability is of paramount importance to any navigation filter, a square-root consider formulation is adopted. Additionally, the

map feature positions are estimated according to the previously described approximation for terrain-aided navigation using FISST. A number of simulation studies are presented to explore the capabilities of the proposed approximation, and they aim to answer the following questions:

- **Study 1:** What are the relevant performance differences between the adopted approximation and the Rao-Blackwellized SMC approaches?
- **Study 2:** How does the performance of "simple" filters, such as PHD filters, and "complex" filters, such as δ-GLMB filters, compare, and what are their relative computational costs?
- **Study 3:** How sensitive is the adopted approximation to the initial sample volume?
- **Study 4:** How is the performance of this terrain aiding concept impacted by the number of features within the map and their associated uncertainties?
- **Study 5:** How does the method react to changes in measurement noise, detection probability, and clutter rate?
- **Study 6:** Since computational efficiency is of paramount importance to onboard navigation, how sensitive is the adopted approximation to GM pruning and merging parameters?
- **Study 7:** How does processing standard, vehicle-specific data within this scheme impact performance?

In each study, 100 Monte Carlo trials are performed, and sample mean and variance are collected to evaluate the results. The variance is used to represent uncertainty in certain quantities, and it will always be presented as a 1σ interval (computed as the square root of the variance). A number of measures are presented to illustrate the performance of the method. These include what are referred to as "mapping performance," quantified using the OSPA metric and cardinality statistics, and "navigation performance," used to describe the errors in position and velocity estimation. Some of these studies involve performing parameter sweeps (such as in Study 5, where values of probability of detection are varied),

but a nominal configuration is defined presently. Unless otherwise specified, all of the studies utilize this set of nominal parameters, and they serve as a baseline to compare results. Again, unless specified here, all other system design and parameters are the same as described in Section 3.2.6.

Let the vehicle state to be of the form $\mathbf{x}_k = [x_k \;\; \dot{x}_k \;\; y_k \;\; \dot{y}_k]^T$, the consider parameter vector to be of the form $\boldsymbol{c}_k = [\beta \;\; b_{\text{cam}}]^T$, and the map states to be of the form $\boldsymbol{\zeta} = [x_\zeta \;\; y_\zeta]^T$. The initial mean and square-root factor are the same as they were before, and the consider parameter vector is taken to have mean and square-root factor $\boldsymbol{m}_{c,k} = [m_\beta \;\; 0]^T$ and $\boldsymbol{S}_{cc,k} = \text{diag}\{\sigma_\beta, 2\}$ for all k, respectively, where, as before, m_β and σ_β^2 are the mean and variance of the uniform density $U(\beta_L, \beta_U)$ with $\beta_L = 10,000$ kg/(m-s^3) and $\beta_U = 63,000$ kg/(m-s^3). The process noise parameter is set to $q = 1$. To simulate the effects of mapping errors, on each trial, the initial map is comprised of a single Gaussian component per feature with the same initial spherical uncertainties of 50 m (1σ) and a mean sampled from a Gaussian with this uncertainty centered upon the true map locations. That is, the true map is the same for all of the Monte Carlo trials and a given trial initializes a map by sampling from the true map to simulate mapping errors. The measurement noise standard deviation is set to 2 pixels, and the probability detection is nominally set to 0.95. In most of the studies, the number of trajectory samples drawn upon initialization is set to $N = 100$, component merging is performed according to a 90% agreement test, pruning is performed such that components with weights below 1×10^{-5} are removed from consideration, and the OSPA cutoff and order parameters are set to $c = 250$ m and $p = 2$, respectively.

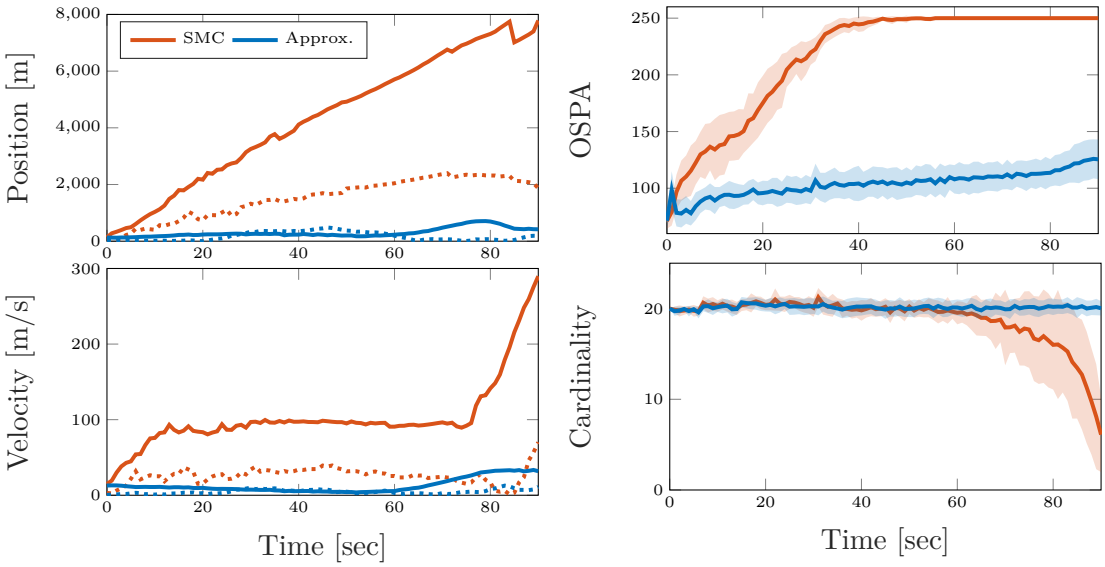
A reader will find that this nominal set of parameters provides desirable performance for this scenario with regard to a balance between navigation performance, mapping performance, and computational burden. So, these studies permit insight into the proposed method, but they also serve as a trade study to determine an ideal parameter selection. Of course, quantities such as initial state uncertainties, mapping errors, measurement noise, etc. are not model parameters to simply choose, but they are immutable components of a given system. The initial state uncertainty is the product of some initialization process, and the observation errors are products of a selected camera and image processing scheme. These are not parameters to tune within a filter until desired performance is achieved, and

are, instead, facts of the stochastic processes that the filter must cope with. Nevertheless, the effects of changes in these quantities is investigated to illuminate any resulting sensitivities.

Study 1: Comparison of Approximation and SMC Methods. To illustrate the salient differences between the proposed approximation (as shown in Figure 4.8) and the previously alluded to SMC methods, a PHD filter using the proposed approximation is implemented, and the Rao-Blackwellized PHD-SLAM technique described in [102, 110] is applied to this problem using the so-called single-feature strategy. The single-feature strategy relies on assuming the map contains only a single feature such that the multitarget likelihood function required to update the weights of the trajectory samples can be evaluated in closed form. Here, as suggested in the provided references, the feature used to evaluate the multitarget likelihood is the map feature with the highest single-target likelihood weighting at the current reference epoch. Both the approximate and SMC methods employ an initial sampling of $N = 100$ trajectory samples, and the SMC method utilizes this same sample density throughout. As is prudent for any SMC implementation, resampling is implemented to avoid sampling degeneracy, and the expected *a posteriori* estimator of [110] is used to obtain vehicle state and map estimates.

The results of this study are presented in Figure 4.15. The position and velocity statistics are shown in the left column, where the Monte Carlo mean errors are dotted lines and the 1σ interval are solid lines. The statistics of the OSPA metric and cardinality estimate provided by the filter are shown in the right column, where a solid line denotes the mean path and the shading indicates a 1σ interval about that mean. Inspection of these results immediately indicates that the approximate method substantially outperforms the SMC method by several times in both position and velocity, and it appears clear that the SMC method is unable to successfully maintain or improve state knowledge of the vehicle. By contrast, the proposed approximation achieves position and velocity 1σ statistics of approximately 420 m and 32 m/s, respectively. Additionally, the SMC method exhibits very poor mapping performance, as exemplified by its corresponding OSPA metric saturating the 250 m cutoff parameter. By inspecting the cardinality estimate, the effects of the aforementioned challenging problem geometry (targets "stacking" from the vehicle's perspective)

are extremely apparent, where, starting at about 50 seconds, the SMC method's cardinality estimate begins to plummet. What's more, the relative runtimes of the two methods are presented in Figure 4.16, where the presented runtimes are the average trial computing times over all 100 trials and they are normalized such that the maximum runtime is equal to one. This indicates that the proposed approximation only requires 1% of the computational time since it is able to perform model reduction through time.



(a) State error 1σ (solid) and mean (dotted).

(b) OSPA metric and cardinality estimate.

Figure 4.15. Study 1 results.

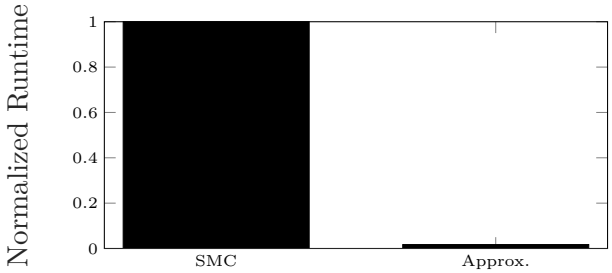
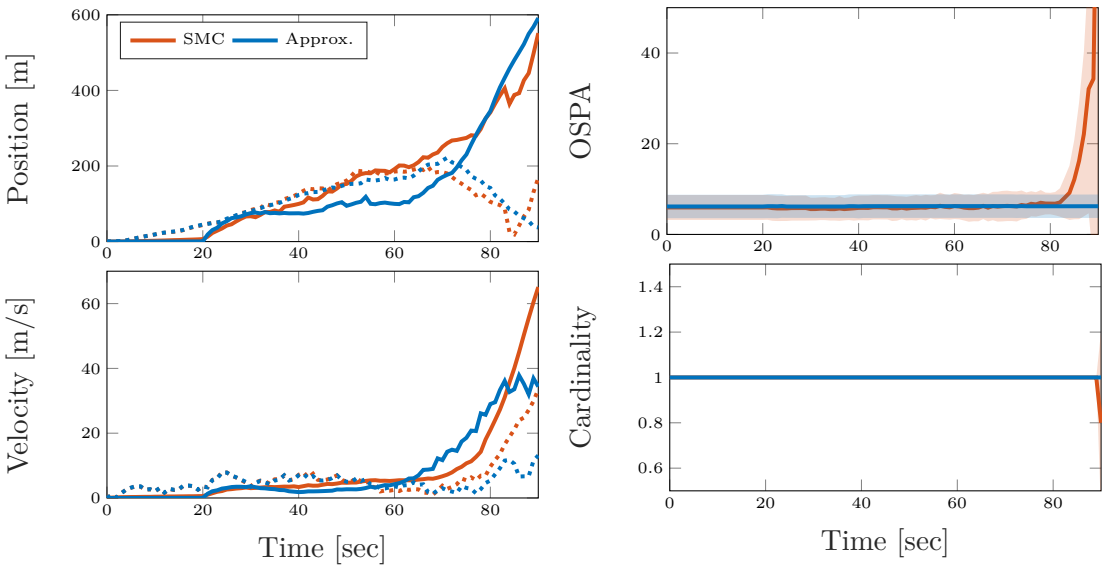


Figure 4.16. Normalized runtimes for the SMC and approximate methods.

At first, it may appear suspect that the proposed approximation not only drastically outperforms the SMC method but also requires only a small percentage of the computing time. Indeed, the author felt such misgivings upon producing these results, so additional analysis was performed to make sense of the results. Consider now the results of Figure 4.17,



(a) State error 1σ (solid) and mean (dotted).

(b) OSPA metric and cardinality estimate.

Figure 4.17. Study 1 results for unrealistically accurate initial conditions.

wherein the same trials are repeated but for initial state uncertainties that are three orders of magnitude lower, to 1 m position and 0.1 m/s, and initial map uncertainties that are reduced to 5 m for a single target. Now, though the SMC method still exhibits a small amount of mapping degradation toward the end of the trajectory, the two methods perform very similarly. This provides one of the key explanations as to why the SMC performs so poorly: the collected samples are insufficient to describe the evolution of uncertainty for this problem, particularly due to the necessarily large state uncertainties. Indeed, this is logical, since 100 samples is clearly a drastic undersampling of the problem, but the proposed approximation appears to perform rather well with the same amount of samples.[11] In practical application, a navigation system cannot rely on being initialized with the unrealistically small state uncertainties seemingly required by the SMC method for these types of problems. Furthermore, it is obvious in this case that the single-feature strategy required to implement the SMC method may be a very poor approximation; it is noted that recent work has proposed improved strategies, such as in [111], but those are not explored here.

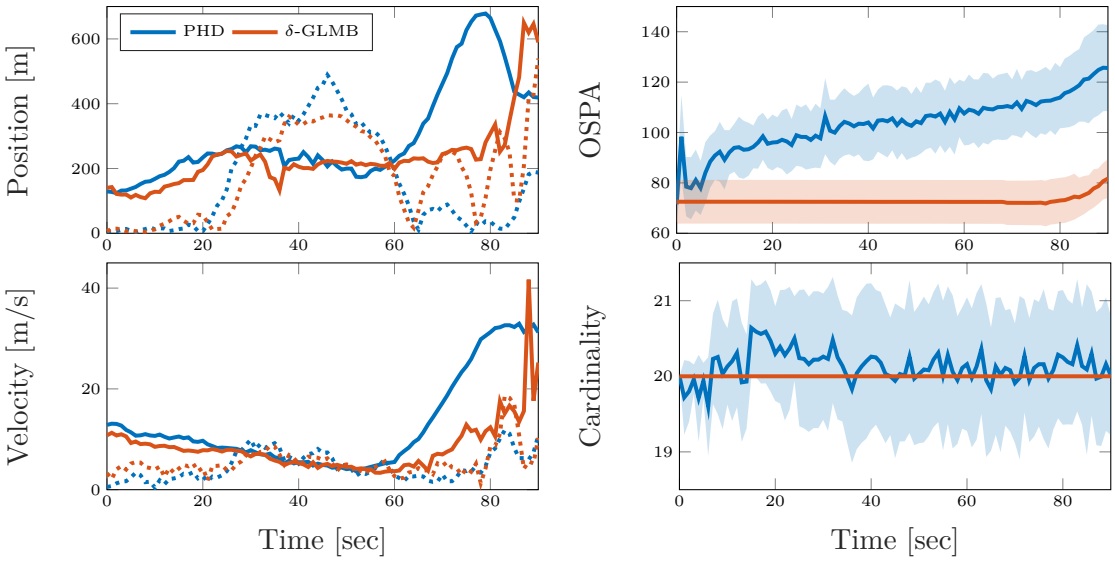
[11] Increasing the number of samples by an order of magnitude showed an improvement in the SMC method, but it was only marginal, and the computational burden skyrocketed.

Conclusions of Study 1: For navigation problems with large initial state uncertainties and appreciable mapping errors, the proposed approximation provides much more desirable all-around performance than the SMC method.

Study 2: Comparison of "Simple" and "Complex" Filters. Since there are so many tools available for multitarget tracking, it is natural to be curious about how one performs in comparison to another for the proposed navigation concept. As described before, to develop conclusions without implementing every one of the vast number of FISST-based filters developed over the years, this work uses a "bookended" approach to illustrate relevant differences. In this case, the bookends are the "simple" PHD filter and the "complex" δ-GLMB because they vary vastly in theoretical, implementational, and (to a varying degree) computational complexity. The PHD filter is configured nominally, but the δ-GLMB filter requires a few additional parameters to be defined to utilize the efficient implementation described in [73]. In particular, when the survival and detection hypotheses are constructed, only the top 50 of each are generated for processing, the maximum number of posterior hypotheses is always capped at 500, and any hypothesis with weight less than 1×10^{-3} is removed from consideration.

The results of this study are shown in Figure 4.18. It is immediately apparent that the mapping performance of the δ-GLMB filter substantially outperforms that of the PHD filter, an unsurprising result since the δ-GLMB filter is known to be a much more effective multitarget tracker. In particular, the δ-GLMB filter maintains the map in a very steady fashion, whereas the PHD filter exhibits a constant, albeit slow, growth in OSPA metric as map estimation and cardinality errors propagate into later filtering solutions. Note that the cardinality estimates provided by the PHD filter have a much higher variance than the other filter; in fact, the δ-GLMB perfectly estimates the cardinality for all 100 trials. Interestingly, both filters exhibit the same increase in OSPA metric toward the end of the trajectory, again due to the observation geometry.

The most striking result is in the position and velocity statistics in Figure 4.18a. In this case, despite their differences in complexity, their vehicle estimation performance is very similar. While it may be claimed, and rightly so, that the δ-GLMB filter somewhat outperforms the estimation performance of the PHD filter, such as at time step $t = 80$



(a) State error 1σ (solid) and mean (dotted). (b) OSPA metric and cardinality estimate.

Figure 4.18. Study 2 results comparing the PHD and δ-GLMB implementations.

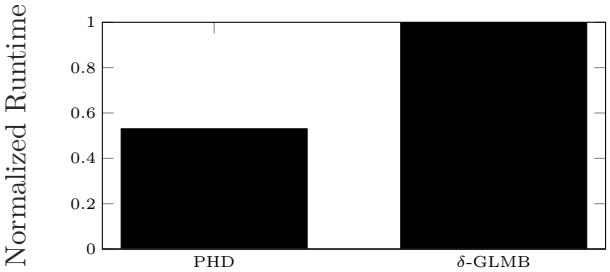
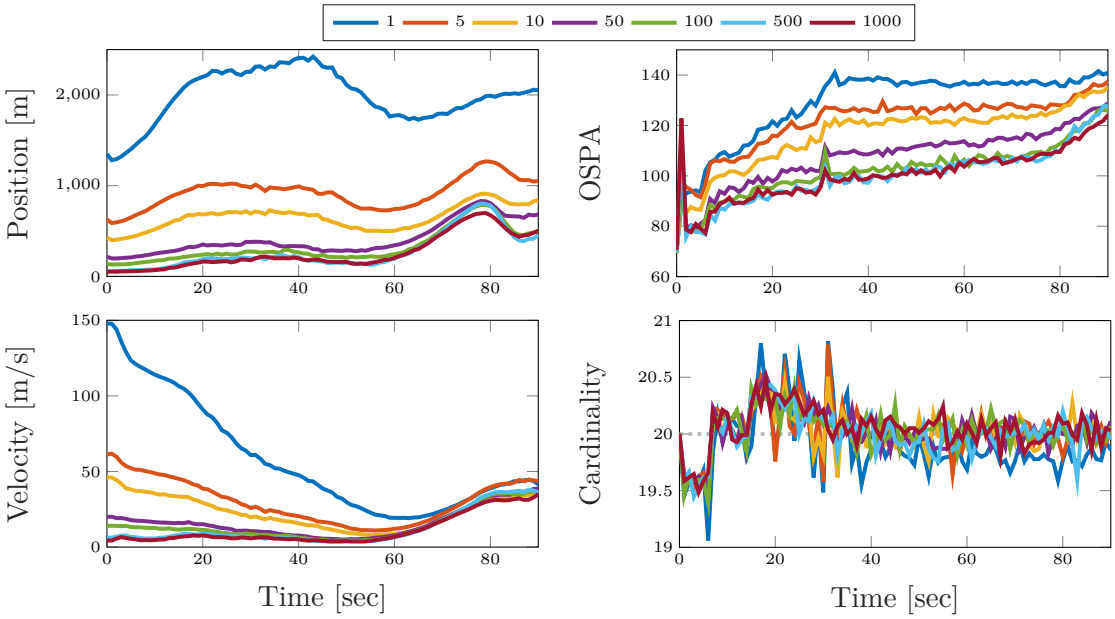


Figure 4.19. Normalized runtimes comparing the PHD and δ-GLMB implementations.

seconds, these differences appear transient and relatively minimal. Furthermore, the δ-GLMB solution appears to have less stability in the 1σ curve for the same number of Monte Carlo trials, indicating that it is producing more widely varying results. Additionally, Figure 4.19 shows the relative runtimes and indicates that the PHD filter only requires half the computing time of the δ-GLMB filter. Therefore, given a collection of mission requirements, a navigator can select the PHD filter, the δ-GLMB filter, *or* another filter somewhere in-between, based on a mission's mapping requirements without fear of large differences in vehicle state estimation.



(a) State error 1σ interval.

(b) Mean OSPA metric and cardinality.

Figure 4.20. Study 3 results comparing the initial sampling volume.

Conclusions of Study 2: The PHD and δ-GLMB filters perform very similarly with respect to navigation performance despite their complexity differences, and their significant differences lie in mapping performance.

Study 3: Initial Sampling Sensitivity. The approximate method relies upon an initial sampling to characterize the initial vehicle state uncertainty, and the previous studies have employed $N = 100$. To investigate the sensitivity of this method, N is varied between 1 and 1,000 for the PHD filter, and the performance is presented in Figure 4.20. Due to the sheer density of curves contained in the figure, the dotted means and shaded regions that were shown in Studies 1 and 2 have been omitted.

A convergence trend is immediately observed, and diminishing returns are observed in position, velocity, and OSPA metric after the initial sampling is increased beyond 50. Cardinality estimation appears wholly unaffected by the differences in sampling density. At first, it is tempting to conclude that setting $N = 500$ is a good choice for initial sampling, but Figure 4.21 contains the relative runtimes of the different sampling densities. If, instead, $N = 100$ is selected, nearly identical vehicle state estimation and mapping performance is

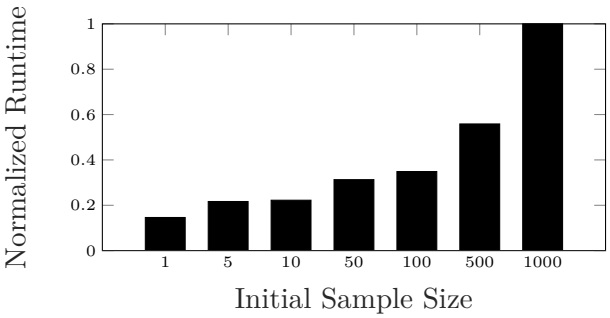


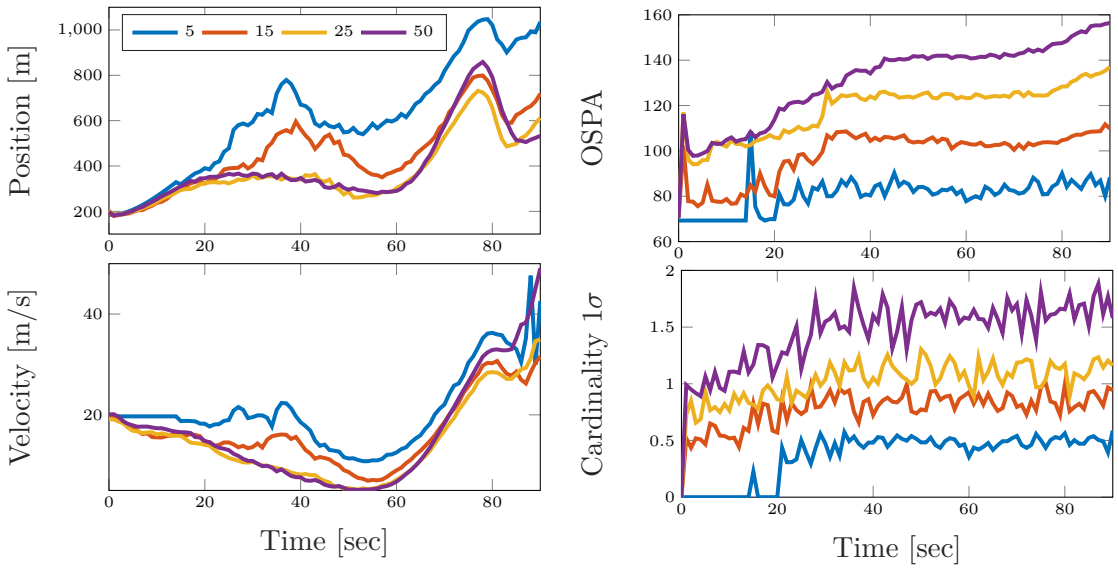
Figure 4.21. Normalized runtimes comparing the initial sampling volume.

obtained, as seen in Figure 4.20, at 40% the cost of initializing with 1000 samples and only 67% the cost of initializing with 500 samples, as seen in Figure 4.21. Decreasing further to $N = 50$ provides meager computational savings but a degraded estimation accuracy and is thus discouraged.

Conclusions of Study 3: Desirable navigation and mapping performance can be obtained with as few as 100 samples at initialization.

Study 4: Map Sensitivity. To evaluate the sensitivity of the proposed method to changes in the initial map provided to the filter, two sub-studies are presented. The first varies the number of features contained by the map, and the second varies the initial uncertainty of the features in the map. Again, the PHD filter is employed.

Varying Map Size. The results of varying the number of features from 5 to 50 are shown in Figure 4.22. Note that, in contrast to previous plots, the 1σ interval collected from the cardinality estimates of the samples is plotted rather than the cardinality itself. This is because, since the PHD filter models targets as Poisson RFS, the variance in cardinality estimates should increase as the map size increases. Indeed, this can be seen in Figure 4.22b, where a larger number of map features results in a larger cardinality variance. Otherwise, it can be inferred that as the number of map features increases, so does the OSPA metric. This is largely because of the cardinality errors induced by the larger true cardinality and is an unsurprising result for a PHD-type filter; high-cardinality estimation is not the PHD filter's strong suit.



(a) State error 1σ interval.

(b) Mean OSPA metric and cardinality 1σ.

Figure 4.22. Study 4 results, varying number of map features.

However, Figure 4.22a shows that vehicle state estimation is almost the same for 15 targets as it is for 50 targets, and nearly identical for 25 and 50 targets. This indicates a somewhat counterintuitive result: in this case, more features do not necessarily translate into better estimation performance. This means that a navigator should take care in designing the map employed by the vehicle's onboard filter, especially when considering the runtime performance in Figure 4.23. The difference in runtime requirements for 25 and 50 targets is drastic, despite their nearly identical vehicle state estimation performance. At first, this is an alarming trend to see for the PHD filter, since the PHD filter is known to exhibit linear complexity in target number. However, it is important to note that this is an analytical conclusion and disregards model reduction techniques such as merging and pruning. It turns out that, in this case, the PHD filter is able to much more efficiently merge and prune for the 25 feature case than the 50 feature case.

Varying Map Uncertainty. The results of varying the initial map uncertainty of the 20 features from 5 m to 250 m are shown in Figure 4.24. The first thing to note is that, entirely unsurprisingly, mapping performance is degraded as initial map uncertainty is increased, as shown by the OSPA metric trends. However, what is of key importance

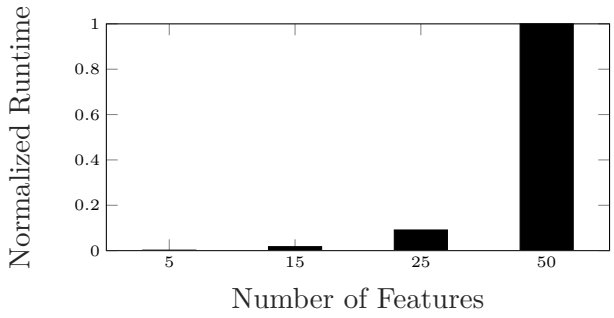
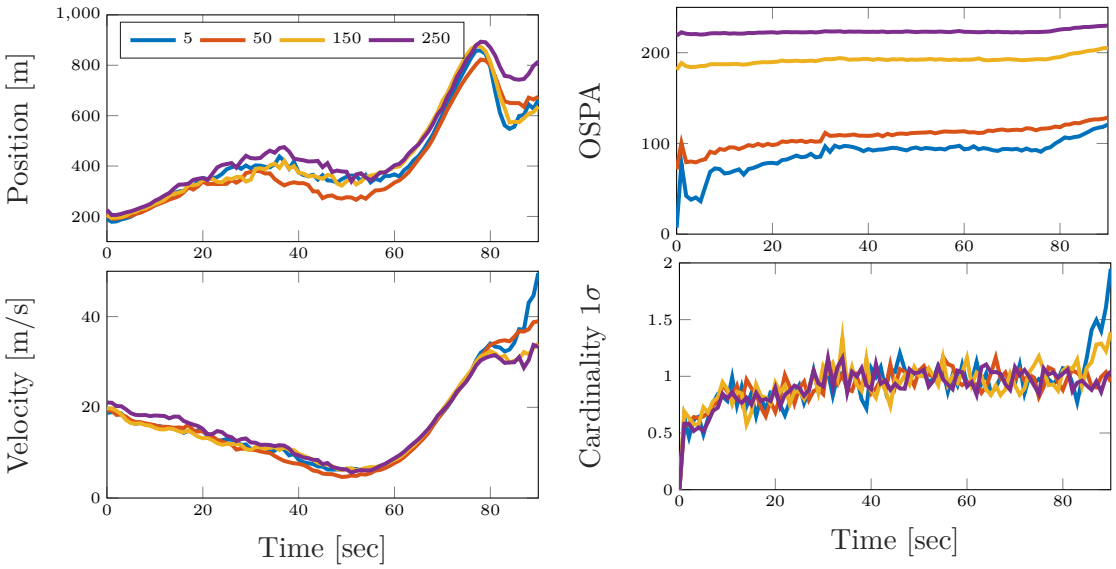


Figure 4.23. Normalized runtimes for varying numbers of map features.



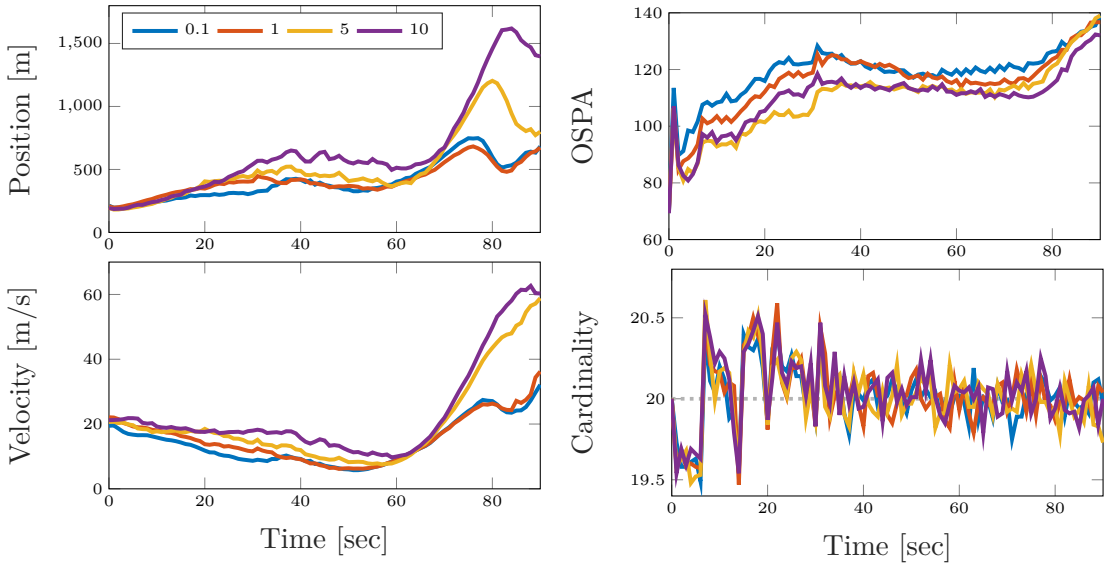
(a) State error 1σ interval.

(b) Mean OSPA metric and cardinality 1σ.

Figure 4.24. Study 4 results, varying initial map uncertainty (in m).

here is that the vehicle state estimation appears nearly unaffected by changes to this map uncertainty, indicating a degree of robustness to mapping errors. Said another way, despite the fact that mapping errors, and associated uncertainties, are increased by a factor of 50, navigation performance is largely unaffected.

Conclusions of Study 4: Diminishing returns are observed as the number of map features increases, indicating that more map features do not necessarily yield improved navigation performance despite the fact that it yields more data. Additionally, the proposed method appears largely unaffected to large changes in initial mapping errors and associated uncertainties.



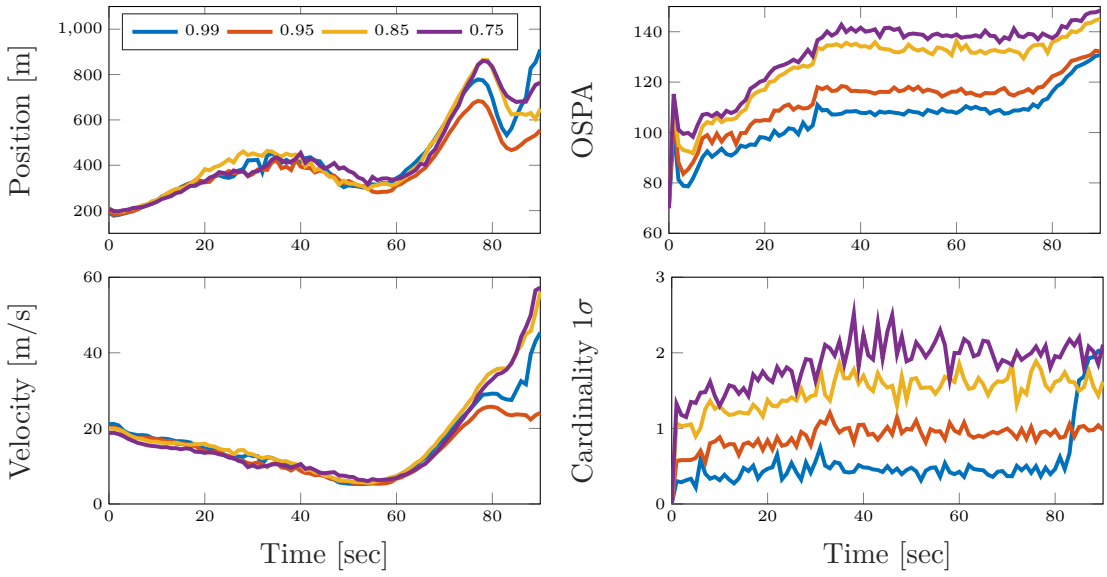
(a) State error 1σ interval.

(b) Mean OSPA metric and cardinality.

Figure 4.25. Study 5 results, varying measurement noise (in pixels).

Study 5: Signal-to-Noise Ratio Sensitivity. To illustrate the proposed approximation's sensitivity to the observational scheme, the parameters controlling the terrain camera are varied, and the performance of the PHD filter is recorded. In particular, three sub-studies are presented: varying measurement noise, varying detection probabilities, and varying the rate of erroneous/clutter returns.

Varying Measurement Noise. The results of varying the camera's measurement noise from 0.1 to 10 pixels is shown in Figure 4.25. Interestingly, this has little impact on the mapping performance. The cardinality estimation is seemingly unaffected, and the OSPA metric, while largely unchanged, actually indicates that the filter conducts mapping more accurately with larger measurement uncertainties. The reason for this is unclear, but it is believed that the larger measurement uncertainties prevents overly-severe updates to the map feature estimates. Of most importance, though, are the trends for position and velocity estimation, where a clear convergence trend is visible and logical results are obtained: more noise yields poorer navigation estimates.



(a) State error 1σ interval. (b) Mean OSPA metric and cardinality 1σ.

Figure 4.26. Study 5 results, varying probability of detection, $p_{D,k}$.

Varying Detection Probability. The results of varying probability of detection from 0.99 down to 0.75 are presented in Figure 4.26, where, again, it is noted that the cardinality uncertainty, rather than the mean, is shown. As expected, as probability of detection decreases, mapping performance degrades, as seen in both the OSPA metric and cardinality uncertainty. The position and velocity estimation performance is surprisingly robust to changes in probability of detection, and it is mostly the high-acceleration, challenging geometry at the later portion of the trajectory that exposes the largest differences. Rather interestingly, the very high probability of detection, 0.99, is outperformed by the lower value of 0.95.

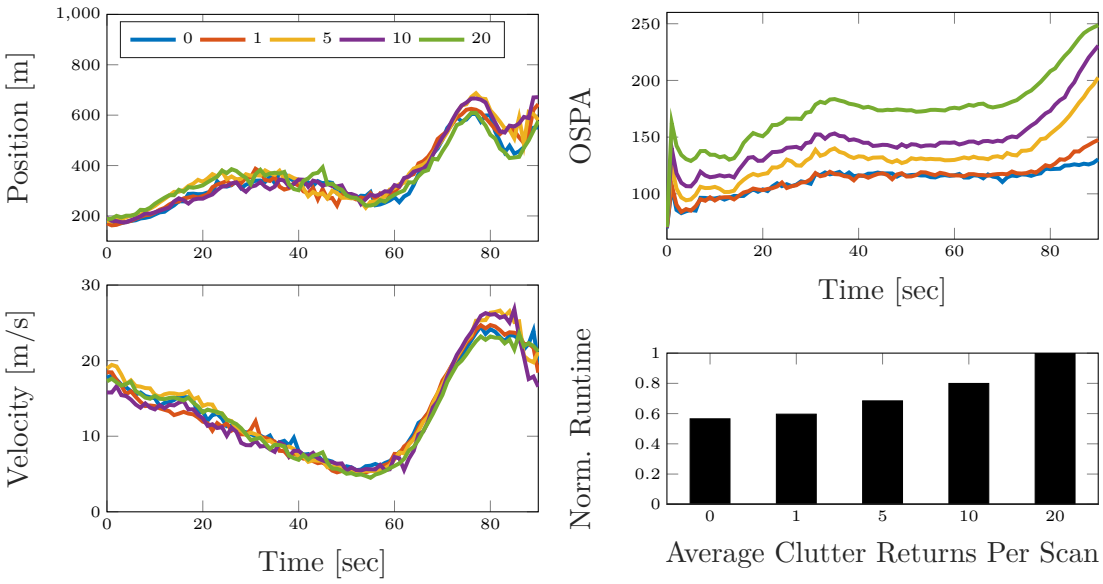
This may seem counterintuitive, since lower probabilities of detection result in less data to process, but it turns out that a slightly lower probability of detection such as 0.95 gives the filter a bit of "breathing room." To understand what is meant by this, recall the PHD filter corrector of Eq. (4.3), repeated here as

$$v_k^+(\bar{\boldsymbol{x}}_k) = [1 - p_{D,k}]v_k^-(\bar{\boldsymbol{x}}_k) + \sum_{\boldsymbol{z}\in\boldsymbol{Z}_k} \frac{p_{D,k}g(\boldsymbol{z}|\bar{\boldsymbol{x}}_k)v_k^-(\bar{\boldsymbol{x}}_k)}{\kappa_k(\boldsymbol{z}) + \int p_{D,k}g(\boldsymbol{z}|\bar{\boldsymbol{x}}_k)v_k^-(\bar{\boldsymbol{x}}_k)\mathrm{d}\bar{\boldsymbol{x}}_k}\,.$$

These two terms work in tandem to account for features that were not and were observed, respectively. If the probability of detection is very high, the filter is able to make very confident claims about the posterior PHD, such as, in the second term, severely down-weighting things it did not observe to be represented in the data that it deemed should have been observed. In theory, if $p_{D,k}$ matches the true detection process, then no problems should arise. Practice, however, is a different story. It is well known that the PHD filter suffers from the so-called "spooky effect" where a missed detection of one target causes an inexplicable increase in mass of another target's PHD. In traditional multitarget tracking, this is highly undesired since it is illogical and degrades tracking performance. In this case, however, this "spooky effect" assists in distributing extra PHD mass around the *vehicle state estimate* portion of the PHD as well, somewhat serving as an underweighting factor within the PHD. So, a slightly decreased detection probability, such as 0.95, and, complementarily, an increased missed detection rate, affords the navigation filter some robustness to severe updates by the PHD filter.

Varying Clutter Density. The results of varying the average number of clutter returns per scan from 0 to 20 are shown in Figure 4.27. As expected, the mapping errors, represented by the OSPA metric, increase as the clutter rate increases. The vehicle state estimation, however, is seemingly unaffected by changes to clutter rate, indicating that this method is desirable for cluttered scenarios, such as applications where poor or dynamic lighting conditions result in unpredictable image processing. Figure 4.27 also contains the normalized runtimes for this experiment. The PHD filter is known exhibit linear complexity in the number of measurements, as well as number of targets, and these results corroborate that theoretical result.

Conclusions of Study 5: As measurement noise is increased, mapping errors are increased and the navigation performance is degraded. Logically, mapping performance degrades as probability of detection is decreased, but navigation performance appears to suffer if probability of detection is too high. As clutter rate increases, so do the mapping errors, but navigation performance is largely unaffected.



(a) State error 1σ interval.

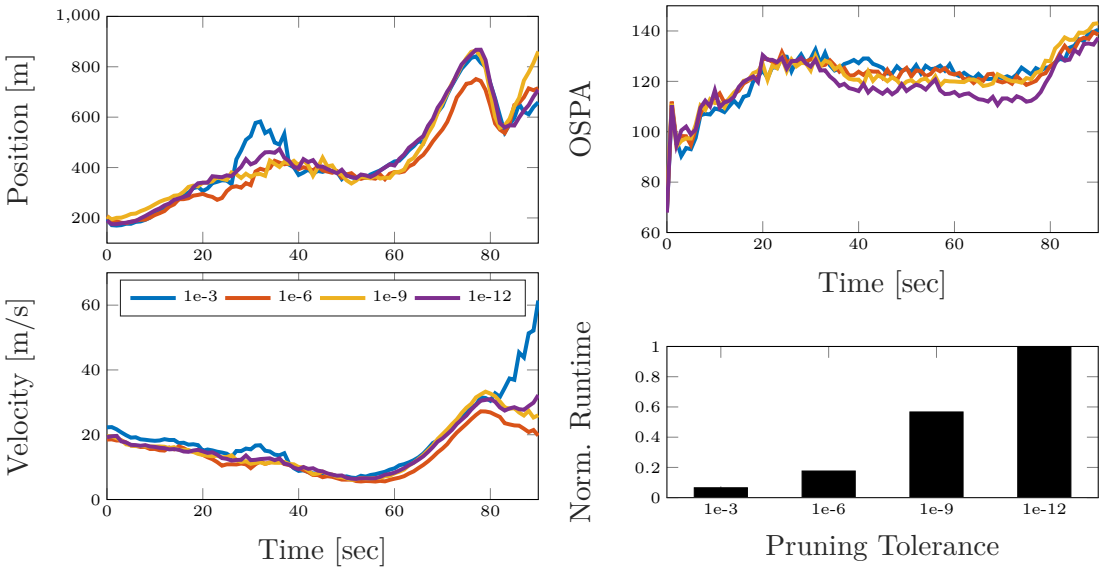
(b) Mean OSPA metric and norm. runtimes.

Figure 4.27. Study 5 results, varying the clutter rate, λ, in average number of clutter returns per scan.

Study 6: GM Maintenance Sensitivity. It has been noted several times that complexity reduction techniques, such as pruning and merging, are necessary for enabling practical implementation of the studied techniques. Here, the sensitivity of the proposed approximation using the PHD filter is investigated by varying the parameters that determine how aggressively the posterior PHD is pruned and merged.

Varying Pruning Tolerance. Recall that pruning refers to removing low-weighted GM components from the posterior PHD. This must inherently result in a finite truncation error, where lower pruning tolerances should result in lower truncation errors, and this sub-study aims to quantify the impact that error has on performance. Pruning certainly enables reduced runtimes, but does it cause the filter's estimation performance to degrade substantially?

The results of varying the pruning tolerance from 1×10^{-3} (aggressive) to 1×10^{-12} (gentle) are shown in Figure 4.28. Note that compact engineering notation is employed in these results, i.e. 1×10^{-6} is shown as 1e-6. The left column indicates that navigation performance is effectively unchanged unless very aggressive pruning strategies are utilized,



(a) State error 1σ interval.

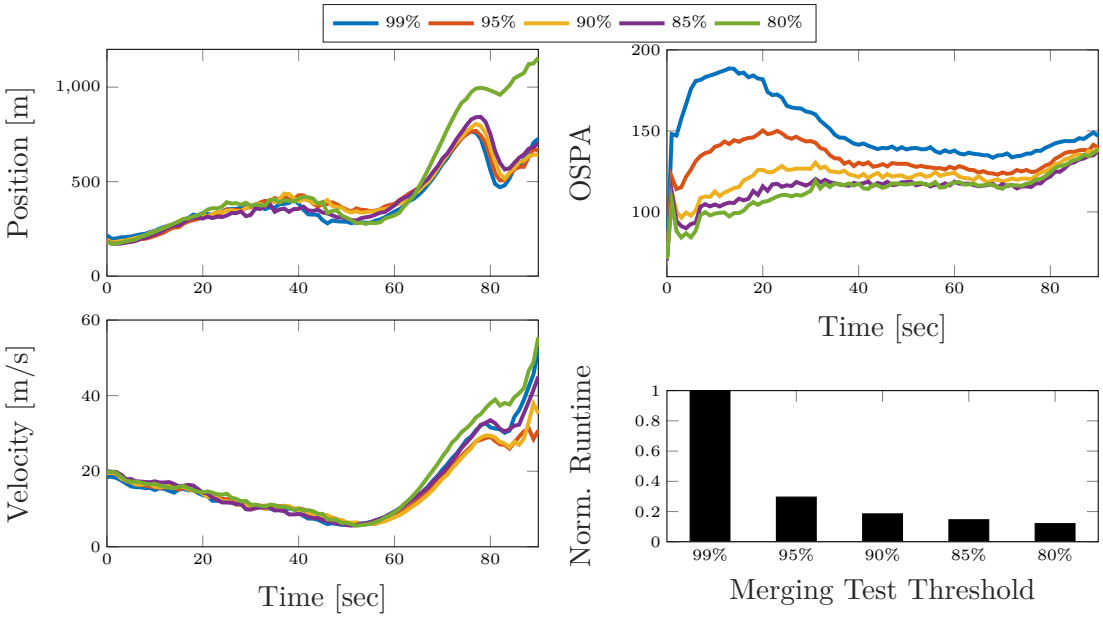
(b) Mean OSPA metric and norm. runtimes.

Figure 4.28. Study 6 results, varying the component pruning tolerance.

such as using a tolerance of 1e-3. Interestingly, mapping performance is effectively unaltered by the changes in the pruning tolerance, though a modest improvement is obtained for the least aggressive of the pruning tolerances, 1e-12. Despite these small differences, the differences in normalized runtime are drastic, and the performance obtained by pruning at 1e-6 is nearly the same as pruning at 1e-12 for only 20% of the computational cost.

Varying Merging Tolerance. Merging is a different approach to GM maintenance than pruning, and recall that, rather than truncating components, it combines statistically similar components into a single-component approximation based on a %-agreement test based on χ^2 statistics. If one sets the %-agreement threshold too low, clearly dissimilar components will be merged, and estimation errors will be induced. This sub-study aims to quantify the effects of changes in this threshold to, as before, trade between performance and runtime.

The results of varying the merging threshold from 99% to 80% are presented in Figure 4.29. The first thing of note is that for all but the most extreme merging threshold of 80%, navigation performance is nearly identical across the parameter sweep. This indicates that relatively aggressive pruning can be performed with little loss of navigation perfor-



(a) State error 1σ interval.

(b) Mean OSPA metric and norm. runtimes.

Figure 4.29. Study 6 results, varying the component merging tolerance.

mance. Counterintuitively, the OSPA metric trends indicates that as merging becomes more aggressive (i.e. the %-agreement threshold drops), mapping performance is actually *improved*. The reason for this is actually explained by the state estimation procedure used to obtain state estimates. In theory, merging should induce an approximation error and, subsequently, create mapping errors. However, when map estimates are obtained, the peaks in the posterior GM are required, and a more aggressively merged map has peaks that are more distinct. Thus, the 80% case outperforming the 99% case is not necessarily indicating that it has a more accurate map. Rather, it indicates that more accurate map feature estimates are able to be obtained due to the more aggressively merged map.

The bottom right panel of Figure 4.29 indicates that merging offers tremendous advantages with respect to runtime, but the advantages exhibit diminishing returns. In this case, it appears that a merging tolerance near 90% offers an agreeable balance between navigation performance, map estimation, and required runtime, without being so low that the navigation performance is corrupted (such as in the 80% case).

Conclusions of Study 6: The proposed approximation exhibits insensitivity to pruning threshold and runtime is substantially reduced as a result, but pruning too aggressively can degrade performance somewhat. Moderately aggressive merging produces desirable reductions in runtime and actually improves mapping performance by simplifying map feature estimation. Merging too aggressively degrades navigation performance without any runtime advantages over slightly less aggressive strategies.

Study 7: Vehicle- and Map- Specific Data. As described in Section 4.3.3 on pp. 169, the proposed approximation is not at all restricted to only processing the map-specific data from the camera, and, in fact, can also process vehicle-specific data. To investigate this possibility, assume that the vehicle is equipped with a barometric altimeter that produces noisy measurements of altitude, i.e.

$$h^{(\text{veh})}(\mathbf{x}_k) = \sqrt{x_k^2 + y_k^2} - r_0 + v_k^{(\text{veh})},$$

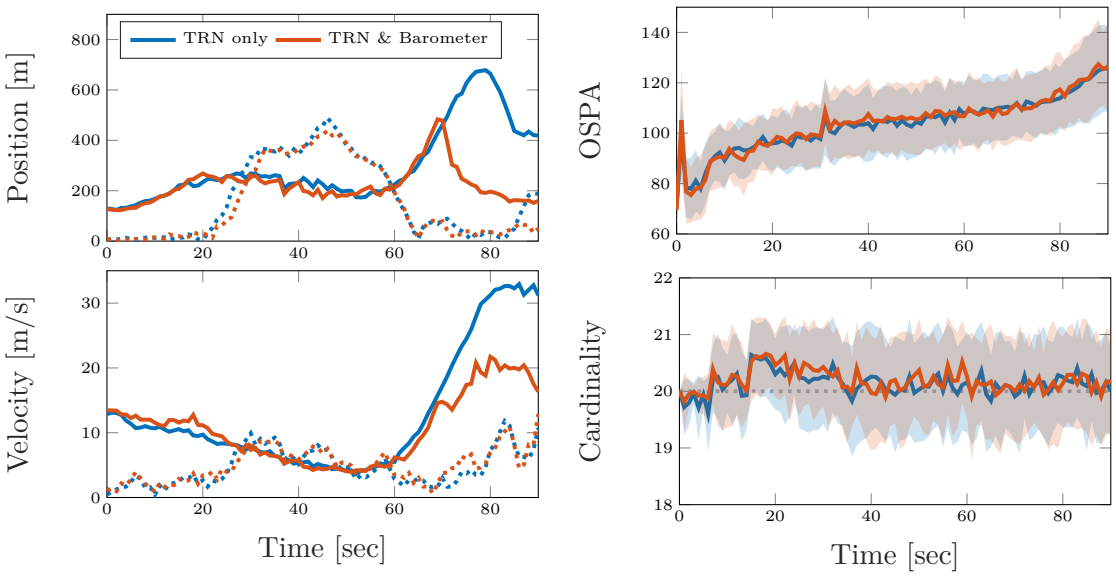
where superscript "(veh)" is appended to distinguish terms from the camera model, r_0 is the local planetary radius, $v_k^{(\text{veh})}$ is a zero mean, Gaussian white noise with standard deviation 500 m, and, as before, $\mathbf{x}_k$ denotes the vehicle state. The corresponding Jacobian with respect to the vehicle-map state $\boldsymbol{x}_k$ can be found to be

$$\boldsymbol{H}_{x,k}^{(\text{veh})} = \left[\frac{\partial h^{(\text{veh})}(\mathbf{x}_k)}{\partial \mathbf{x}_k} \quad \mathbf{0}_{1\times n_\zeta}\right] = \left[\frac{x_k}{\sqrt{x_k^2+y_k^2}} \quad \frac{y_k}{\sqrt{x_k^2+y_k^2}} \quad \mathbf{0}_{1\times 2} \quad \mathbf{0}_{1\times 2}\right].$$

The barometric altimeter is said to begin producing altitude measurements 70 seconds into the trajectory, when the density is large enough to be appreciable, and these observations are processed as proposed in Section 4.3.3 on pp. 169.

The results of processing both the terrain camera and barometric altimeter data is compared to using only the terrain camera data in Figure 4.30, where "TRN" is used to denote the terrain camera processing. Intuitively, processing the additional datatype improves the state estimation performance in both position and velocity once the barometric altimeter turns on at 70 seconds, and, since the barometric altimeter data is not dependent

upon the map, mapping performance is effectively unaltered. The emphasis here is that vehicle-specific and map-related data can be processed within a common architecture using the proposed approach.



(a) State error 1σ (solid) and mean (dotted).

(b) OSPA metric and cardinality estimate.

Figure 4.30. Study 7 results for processing both map-related (TRN) and vehicle-specific (barometric altimeter) data within the proposed architecture.

As an additional observation, the navigation results for a case where terrain camera data is entirely unavailable, such that the filter only processes barometric altimeter measurements, are presented in Figure 4.31. In this case, the filter is initialized just as before, and the only difference is that no terrain camera data is processed. Here, it can be seen that since the filter must propagate for a very long time before the barometric altimeter produces data, the filter tends to apparently diverge. Note that this divergence is not caused by numerical errors in covariance computations, because the square-root consider formulation produces estimates with positive definite covariance matrices at all time steps for all of the trials. Instead, the very large state uncertainties in the *a priori* distribution when barometric altimeter measurements become available disallows the filter from converging upon an accurate state solution, principally due to linearization errors induced by the large

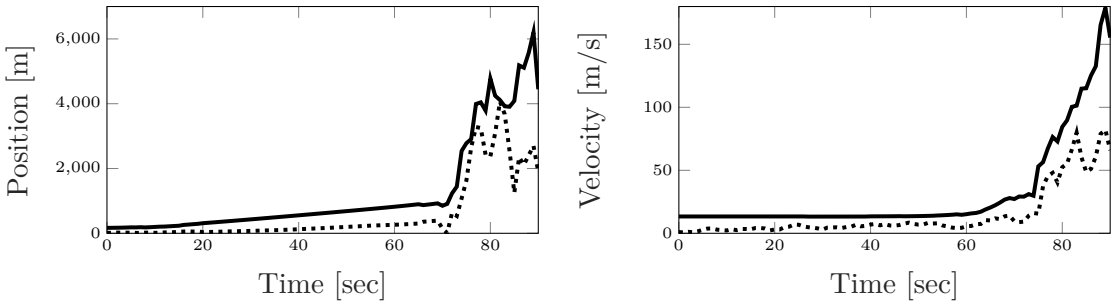


Figure 4.31. State error 1σ (solid) and mean (dotted) for the barometric altimeter only case.

a priori uncertainty. By contrast, inclusion of the terrain camera data permits this growth of uncertainty to be largely abated, and the barometric altimeter data can be subsequently processed to obtain desirable navigation performance.

Conclusions of Study 7: The proposed approach accommodates both map-related and vehicle-specific data to be processed within a common filter.

4.4. SYSTEM AUGMENTATION WITH DECENTRALIZED FUSION

The outlined methods allow terrain camera data to be processed in a FISST-based filter to jointly estimate the vehicle and map states via the approximation in Eq. (4.11). This work further proposes the use of decentralized fusion between the navigation solution produced by the standard approach (such as the EKF) and a SLAM-based navigation solution (such as those based on the PHD or δ-GLMB filters, for example). By doing so, not only can the benefits of both approaches be utilized via a conservative fusion rule, but a fusion implementation can be added to a standard navigation filter with virtually no modification of the existing architecture.

This "augment rather than replace" paradigm produces vastly improved estimation performance without discarding existing investments into well-tested navigation schemes. Principally, this is motivated by pragmatism. A decentralized fusion methodology is useful because many vehicle architectures, particularly those involving space vehicles, are firmly tied to successfully flown, heritage algorithms. It is likely that managers would look skepti-

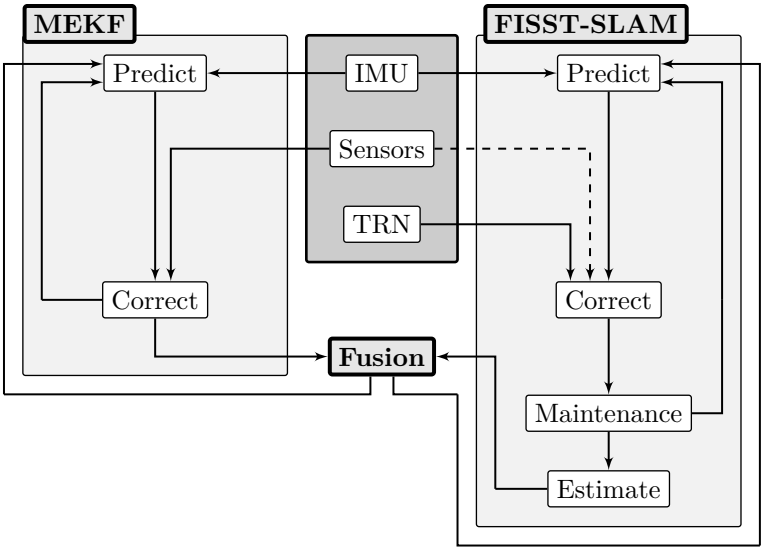


Figure 4.32. Schematic depiction of a FISST-based SLAM fusion implementation augmenting a standard MEKF algorithm.

cally upon adopting an entirely new navigation scheme in lieu of traditional Kalman filtering techniques that successfully put boots on the Moon. Indeed, these missions are remarkably expensive, and risk management is crucial to ensuring success! Instead, a "bolt on" approach such as this one provides potential for state-of-the-art, emerging techniques to accompany traditional, well-tested navigation architectures in a less intimidating fashion.

This scheme is presented schematically in Figure 4.32, where the multiplicative extended Kalman filter (MEKF) [37, 112] is augmented with the new, FISST-based SLAM methodology using fusion with feedback. Here, "TRN" represents terrain-relative navigation and is used to denote a sensor, such as a terrain camera, that collects terrain-related data. This figure makes it clear that the MEKF and FISST-SLAM mechanisms are fed data from shared sources, such as the IMU, and this motivates the use of a conservative fusion rule that is guaranteed not to double count information.

It is important to remember that the SLAM formulation of this work offers the benefit that vehicle-specific sensors, such as an altimeter or a star camera, can be processed directly via suitable definition of the measurement functions. This capability is designated by the dashed line in Figure 4.32. So, if augmenting a standard navigation filter with fusion

is not desired, one could simply process the extra vehicle-specific data entirely within the multitarget filter (as described in Section 4.3.4) in lieu of processing it using an MEKF and fusing the resulting estimates.

4.4.1. Conservative Fusion. Here, it is sought to fuse two probabilistic representations of the vehicle state. Let $p(\mathbf{x}_k|\boldsymbol{Z}_k^{(1)})$ be the pdf produced by the standard navigation approach using all feature-unrelated data $\boldsymbol{Z}_k^{(1)}$, and let $q(\mathbf{x}_k|\boldsymbol{Z}_k^{(2)})$ be the pdf produced by the SLAM filter as a result of processing data $\boldsymbol{Z}_k^{(2)}$. It is natural to first turn to Bayes' rule when considering fusion of two probabilistic sources, given as

$$p(\mathbf{x}_k|\boldsymbol{Z}_k^{(1)} \cup \boldsymbol{Z}_k^{(2)}) = \frac{p(\mathbf{x}_k|\boldsymbol{Z}_k^{(1)})q(\mathbf{x}_k|\boldsymbol{Z}_k^{(2)})}{p(\mathbf{x}_k|\boldsymbol{Z}_k^{(1)} \cap \boldsymbol{Z}_k^{(2)})}\,,$$

but it is immediately apparent that complete knowledge of the common information, $\boldsymbol{Z}_k^{(1)} \cap \boldsymbol{Z}_k^{(2)}$, must be available and explicitly accounted for. This is clearly a problem, as both the standard and SLAM approaches utilize the same IMU data for the time update! Instead, a rule that is guaranteed not to double count information contained within the pdfs is desired. This work discusses two common conservative fusion rules that are guaranteed not to double count information: the geometric mean density (GMD) and covariance intersection (CI), a special case of GMD.

Geometric Mean Density. The GMD fusion rule combines a collection of densities (here specialized to a pair of densities) via a weighted exponential product of the form [113]

$$\tilde{p}(\mathbf{x}_k) \propto p^{\alpha}(\mathbf{x}_k|\boldsymbol{Z}_k^{(1)})q^{1-\alpha}(\mathbf{x}_k|\boldsymbol{Z}_k^{(2)})\,,$$

where $\alpha \in [0,1]$ is a weight that determines the relative strength of fusion contribution from each pdf. The GMD has many desirable properties, such as being a conservative and effective (i.e. has the potential to improve knowledge) fusion rule. For further details, see [5, 113]. Implementation requires explicit computation of a normalization factor and, for many pdf representations (such as GMs), convenient, closed-form solutions do not exist.

Therefore, this fusion procedure requires sample-based approximation and numerical expectation-maximization (EM) procedures to produce a suitable GM representation for subsequent processing [5, 114]. For a state dimension of a practical problem, the number of samples required to accurately describe the candidate densities drastically increases due to the *curse of dimensionality*, and the cost of the iterative EM procedure aggressively scales with the number of input samples and number of mixands.

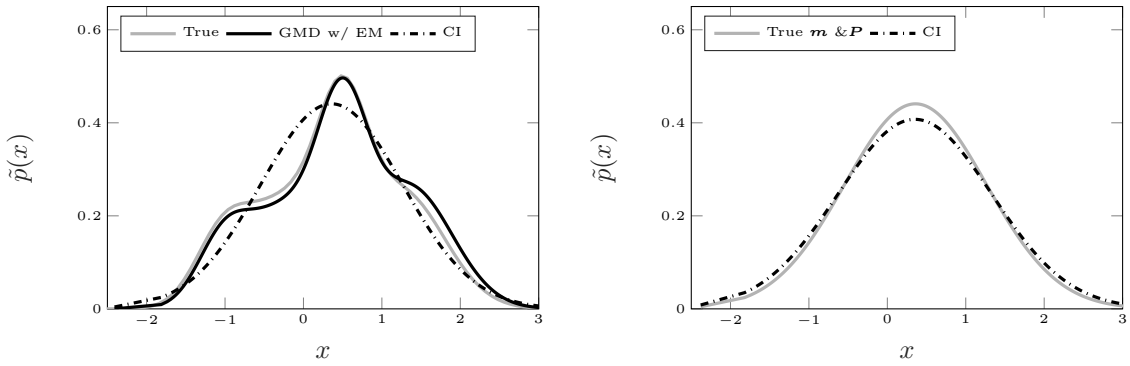
Covariance Intersection. Instead, in the interest of numerically expedient application, if the densities $p(\cdot)$ and $q(\cdot)$ are taken as Gaussian pdfs, GMD specializes to the CI algorithm and admits a convenient closed-form fusion rule. Take the mean $\boldsymbol{m}_k$ and covariance $\boldsymbol{P}_k$ provided by the standard approach's MEKF algorithm, and assume that they parameterize a Gaussian describing the vehicle state of the form $p(\mathbf{x}_k|\boldsymbol{Z}_k^{(1)}) = p_g(\mathbf{x}_k\,;\,\boldsymbol{m}_k, \boldsymbol{P}_k)$. Additionally, let the estimate produced by the SLAM algorithm be Gaussian with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$ such that they parameterize the Gaussian pdf $q(\mathbf{x}_k|\boldsymbol{Z}_k^{(2)}) = p_g(\mathbf{x}_k\,;\,\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. As discussed in Section 4.3.4, the terms $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ can be computed from the vehicle state portion of the GM density produced by the filter using the method of moments. Then, CI produces the Gaussian pdf $\tilde{p}(\mathbf{x}_k) \approx p_g(\mathbf{x}_k; \tilde{\boldsymbol{m}}_k, \tilde{\boldsymbol{P}}_k)$ as the fused density, where [115]

$$\begin{aligned}\tilde{\boldsymbol{m}}_k &= \tilde{\boldsymbol{P}}_k(\alpha \boldsymbol{P}_k^{-1}\boldsymbol{m}_k + (1-\alpha)\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k)\\ \tilde{\boldsymbol{P}}_k &= (\alpha \boldsymbol{P}_k^{-1} + (1-\alpha)\boldsymbol{\Sigma}_k^{-1})^{-1}\,.\end{aligned}$$

It seems that this may be a very agreeable approximation in the context of vehicle navigation, as it is commonly approximated, and is logically expected, that the true vehicle state density is unimodal and, further, approximately normally distributed. Therefore, this work employs CI as its fused density approximation.[12]

A comparison between the approximate GM fusion rule with EM and CI using the conditional mean and covariance of the fused densities is shown in Figure 4.33. As expected, the approximate fusion with EM more closely represents the true fused density, but the far less expensive CI fusion rule approximates the fused statistics rather well. Looking at Fig-

[12]For methods that use covariance intersection in cases where attitude is part of the vehicle's state vector, see [36].



(a) The true GMD fused density, a GM approximation using EM, and the result of CI.

(b) Visualization of the mean and covariance of the true fused density and the result of CI fusion.

Figure 4.33. Comparison of the discussed fusion techniques.

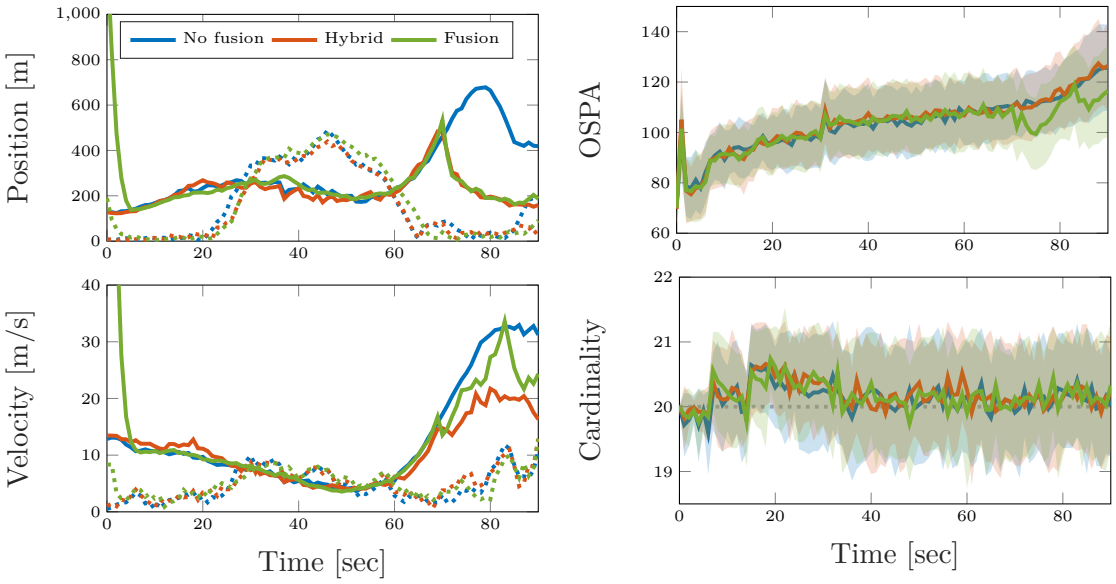
ure 4.33a might cause one to think the CI result is a poor approximation, but Figure 4.33b indicates that the CI fusion rule actually very closely matches the mean and covariance of the true fused density (here plotted according to a Gaussian for visualization). Of course, this is just one case and CI cannot be expected to perform this well in all cases, particularly in highly non-Gaussian environments. Within the context of this dissertation, however, the true vehicle state density is expected to be unimodal most of the time and, thus, CI should serve as a sufficient approximation of the fused density without the relatively immense computational burden imposed by the EM-based procedure.

Fusion Weight Selection. Conservative fusion techniques require the weight α to be determined or provided according to some weight selection criterion. Recent work on related fusion topics in [5] has produced a conclusion that is very useful in this context. Many theoretically disparate weight selection techniques have been compared and, while certain performance advantages present themselves for more complex methods, empirical study indicates that simply selecting $\alpha = 0.5$, i.e. the candidate fusion densities are equally weighted, produces agreeable results at virtually no computational cost. One could, of course, modulate α according to a specific weight selection criterion (as in [5]) or perhaps with respect to navigation-internal health flags (that is, if the MEKF solution is expected to be very accurate because it just received an update from a ground-based orbit determination procedure, one could set α such that more weight is placed on the MEKF density).

Fusion Feedback. Once the two densities are fused, the resulting pdf is used to feed back to both the standard and SLAM approaches. The standard filter replaces its mean and covariance estimates with $\tilde{\boldsymbol{m}}_k$ and $\tilde{\boldsymbol{P}}_k$, and the SLAM filter reinitializes (as described in the previous sections) with $\mathbf{x}_k^{(j)} \sim \tilde{p}(\mathbf{x}_k)$. Note that the rate of fusion/feedback can be designed by the user and that once the filter has reasonably converged, one can feasibly set the number of samples to $N = 1$ in the reinitialization step due to approximate normality of the vehicle state, resulting in drastic savings in computational cost.

4.4.2. Numerical Example. To evaluate the proposed fusion augmentation concept of Section 4.4, consider once again the ballistic trajectory example. Let the proposed approximation be used to process the terrain camera data with the PHD filter, and let the barometer data described in Study 7 of Section 4.3.5 be processed by the linearization-based square-root consider filter described previously in Section 3.2. That is, the vehicle-specific barometer data is processed within a vector-valued single-target filter, and it will be augmented with the proposed terrain aiding strategy with set-valued multitarget tracking using fusion with feedback. The state solutions from both methods are fused using CI with fusion weight $\alpha = 0.5$, both filters are reinitialized with the fused solution, and overall state estimates are taken to be the fused solution.

The results of this fusion scheme are shown in Figure 4.34, and they are compared to processing only the terrain camera data (denoted "No fusion") and processing both terrain camera and barometer data within the same filter (as presented in Study 7 of Section 4.3.5 and denoted "Hybrid"). It can be seen that the initial state solution of the fusion method is initially distant from the others since the square-root consider filter is initialized with a single mean and square-root factor, whereas the others rely on 100 initialization samples. Nevertheless, as soon as terrain camera data begins to be processed, the solution converges to the same as the other two. The navigation performance is nearly identical in position, and resides somewhat between the "No fusion" and "Hybrid" methods in velocity. This difference in velocity can be explained by noting that the linearization errors of a mean and square-root factor method will always be larger than a method using GMs with equivalent overall mean and square-root factor. This is because linearization errors scale with *a priori* uncertainty [53], and the GM components possess "smaller" individual square-root factors



(a) State error 1σ (solid) and mean (dotted).

(b) OSPA metric and cardinality estimate.

Figure 4.34. Demonstration of performance for the proposed fusion augmentation approach ("Fusion"), compared to utilizing only map-related data ("No fusion") and the hybrid approach for processing map-related and vehicle-specific data ("Hybrid").

despite having equivalent overall square-root factor. Mapping performance is largely unaffected by the fusion scheme except when the barometer becomes active, where the mean OSPA actually drops below the other two methods.

The implication of these results is that a navigator does not need to abandon decades of research in traditional estimation techniques, like the Kalman filter, to utilize the proposed terrain aiding concept. Rather, terrain aiding capabilities can be obtained by marrying the new techniques with traditional filters, perhaps even operating on two different processors on a vehicle. Furthermore, if one desires to add terrain aiding capabilities to an existing navigation system design, the scheme can be "bolted on" to the existing design, rather than necessitating an entire redesign.

4.5. ESTIMATING MULTIPLE SETS

Many practical applications involve surveillance regions that contain targets not well-described by a common motion or detection model, and filters that attempt to treat these targets identically can be expected to perform poorly. It is easy to imagine a number of problems wherein targets all move differently, such as a region containing vehicles and pedestrians, and/or are detected differently, such as big red buses being easier to detect than bicyclists. There are certainly ways of treating these targets identically by using computer vision or learning-based classification algorithms. What if, however, this process could be performed in a unified manner within the filter directly?

In the context of terrain-aided navigation, a camera observing boulders or craters may generate returns differently depending on geometry or lighting, and roving vehicles on the surface may cause confusion due to motion within the map. It may be tempting to treat all targets as belonging to the same set, and one might simply treat the stationary map features as dynamic targets with zero velocity. However, how does this account for different probabilities of detection? Furthermore, what if *specific* subsets of the map are of interest, such as preferring to estimate the states of the roving targets rather than the static ones? Or, as a particularly interesting case, what if an extremely dense map is partitioned into two sets, one that is of interest and another that is expected to generate returns but is deemed less important? Since the complexity of most multitarget filters scales directly with target number, could computational complexity be dramatically reduced by such a technique? Could the same navigation performance be obtained with fewer targets estimated by an onboard filter? Could one "focus" more on the precisely known features and expend less effort on the more poorly known elements of an *a priori* map? After all, as discussed in Section 4.3, practical navigation has one priority above all: the vehicle state estimate. These are all questions that this section seeks to investigate.

This is not a newly posed problem, and a number of researchers have proposed a number of solutions to these types of problems. Often, the term "group targets" is applied to cases where "squads" of targets move along a trajectory defined by a group variable. The group variable essentially defines a parent process that generates a number of child

processes that are standard targets, and, therefore, the total collection of targets of all the groups is the total number of targets in the scene. Unified methods were proposed early on [116, 117], works such as Swain's dissertation in [118] explored rigorous techniques for PHD-based solutions to these problems, and work is ongoing, such as Legrand's recent efforts in [119]. Other techniques for such nonstandard targets are PHD methods for jump Markov systems, ultimately aiming to estimate the states of multiple maneuvering targets. Pasha et al. developed a closed-form GM recursion for these problems in [120], and Wei et al. [121] proposed a "PHD-like" filter bank to solve some of the challenges of tracking these types of targets. Different still are "extended targets", where a single target is permitted to generate more than a single measurement (i.e. the target is not treated as a point but as a model-able extent). Due to its immense practical application, substantial research has been dedicated to this topic with PHDs, such as the work of Swain and Clark [122], Swain [118], Orguner et al. [123], Granström et al. [124], and the list goes on. For a thorough review of extended target tracking, see Granström's survey paper in [125].

This is not intended to be a comprehensive presentation of available work. Instead, the point of the preceding discussion is to say that many specialized and varied techniques have been proposed to these types of problems. The distinctions between the published methods become fuzzy since they all, in some fashion, attempt to solve a similar problem but become very specialized to the specific application. All of these techniques–group, jump Markov, extended targets and so on–stem from the same inspiring challenge: targets that require some level of classification. Group target tracking classifies the multiple targets using the group state, the jump Markov techniques classify targets by associating them with labeled and known dynamical models, and extended target tracking essentially classifies observations as belonging to a given target extent versus another.

Granted, the differences between the methods are nuanced, but they are not as different in principle as they may appear. It becomes dizzying when wading through the vast (and growing) amounts of literature, searching for a "common thread" between these methods. Even terminology and nomenclature are subjects of debate within the community, such as in [126], where Mahler rather directly seeks to refute Streit's interpretation of equivalence between FISST and point processes in [127].

Rather than participate in this debate or attempt to formally establish any equivalence between the aforementioned techniques, requirements for a desired filter are designed and the corresponding filter is devised in the following sections. These desired requirements are as follows:

- The filter will estimate a known number of independent random sets, each containing a time-varying, unknown number of targets.
- Targets will not be permitted to transfer from one set to another.
- Each set will be permitted to have distinct motion and detection models.
- The sets will be permitted to belong to different state spaces.
- Each set is estimated using its first-order moment to develop a multiple-set form of the PHD filter.

After some mathematical preliminaries and problem construction, the following discussion will present the predictor and corrector equations for such a filter in Sections 4.5.2 and 4.5.3, respectively; details of a GM implementation are described in Section 4.5.4, an attempt at a summary comparison between the new filter and filters that exist in literature is presented in Section 4.5.5; and, finally, some numerical studies are presented in Section 4.5.6.

Remark 4.5 (On Absent Literature)**.** *To the best of the author's knowledge, the filter that is constructed here is new to literature. Somewhat related tools seem to be mentioned in places, such as Reference 36 in [101], a "variable state space CPHD filter for jump Markov models", that is seemingly the same work as Reference 75 in [128], cited as being submitted to IEEE's Transactions on Aerospace and Electronic Systems but which never appeared. The title of Reference 57 in [129] indicates a "joint intensity filter" but this memo does not appear anywhere. All of these results do not seem to be publicly disseminated or published in archival journals, and therefore any interrelation cannot be accounted for here.*

Some Mathematical Preliminaries. The concept of the probability generating functional (PGFL) of an RFS (or, more precisely, the PGFL corresponding to that RFS's multitarget density) is the cornerstone of much of FISST-based filtering theory, particularly

when developing techniques that utilize the PHD, and the PGFL is a fair analog to the standard probability generating function in traditional statistics. PGFLs are used primarily as a tool to obtain the first-order statistical multitarget moment, or the function called the PHD, and useful results are obtained, as with standard probability generating functions, by taking derivatives. PGFLs will be used extensively to produce the results of the following discussion and, therefore, they are discussed presently. For a much more complete look into PGFLs and their impact on FISST-based filters, a reader is recommended to delve into [82], [83], and [101].

For some RFS $\boldsymbol{Y}$, its PGFL according to some (unitless) test function $h(\boldsymbol{y})$, with $\boldsymbol{y} \in \boldsymbol{Y}$ such that $0 \leq h(\boldsymbol{y}) \leq 1$, is given by the expected value

$$G[h] = \mathbb{E}\left\{h^{\boldsymbol{Y}}\right\} = \int h^{\boldsymbol{Y}} f(\boldsymbol{Y}) \delta \boldsymbol{Y}\,,$$

where the above integral is a set integral and the common shorthand notation

$$h^{\boldsymbol{Y}} = \prod_{\boldsymbol{y} \in \boldsymbol{Y}} h(\boldsymbol{y})$$

is adopted. One can extend this result to an N argument PGFL of N sets $\boldsymbol{Y}^{(1)}, \ldots, \boldsymbol{Y}^{(N)}$ as

$$G[h_1, \ldots, h_N] = \int \cdots \int h_1^{\boldsymbol{Y}^{(1)}} \cdots h_N^{\boldsymbol{Y}^{(N)}} f(\boldsymbol{Y}^{(1)}, \ldots, \boldsymbol{Y}^{(N)}) \delta \boldsymbol{Y}^{(1)} \cdots \delta \boldsymbol{Y}^{(N)}\,. \tag{4.12}$$

As with standard probability generating functions, setting any $h_i = 1$ effectively marginalizes out the corresponding variable for consideration. The principal use of PGFLs in this dissertation is to obtain expressions for PHDs. A density's PGFL, $G[h]$ and its PHD, $v(\boldsymbol{y})$, are related through the set derivative as

$$v(\boldsymbol{y}) = \frac{\delta G}{\delta \boldsymbol{y}}[h]\bigg|_{h=1}\,.$$

That is, if $G[h]$ is the PGFL of a given set, thus providing a full statistical description of that set, its PHD is obtained through set differentiation.

If one considers the set $\boldsymbol{Y}$ such that $\boldsymbol{Y} = \boldsymbol{Y}^{(1)} \cup \cdots \cup \boldsymbol{Y}^{(N)}$, where each $\boldsymbol{Y}^{(i)}$ has some PHD $v^{(i)}(\cdot)$, and defines the joint PGFL in Eq. (4.12), the N^{th}-order factorial moment is found using

$$v(\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}) = \frac{\delta^N G}{\delta \boldsymbol{y}^{(1)} \cdots \delta \boldsymbol{y}^{(N)}}[h_1, \ldots, h_N]\Big|_{h_1=1,\ldots,h_N=1} . \tag{4.13}$$

The PHD is also known as the first-order factorial moment density, such that, for $N = 1$,

$$v(\{\boldsymbol{y}\}) = v(\boldsymbol{y})$$

almost everywhere [87]. Interestingly, if all elements of $\boldsymbol{Y}$ are generated according to a Poisson process, it holds that

$$v(\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}) = \prod_{n=1}^{N} v^{(n)}(\boldsymbol{y}^{(n)}) \tag{4.14}$$

i.e. the N^{th}-order factorial moment is the product of the N PHDs. This result will be referred to as the "PHD-product."

Oftentimes, when taking these types of set derivatives, the derivative of a product is required. While the product rule is a commonplace component of elementary calculus, the general expression for the n^{th} derivative of a product of m terms is a handy expression to keep within arm's reach and is given by

$$\frac{\mathrm{d}^n}{\mathrm{d}x^n}\left\{\prod_{k=1}^{m} f_k\right\} = \sum_{k_1+\cdots+k_m=n} \binom{n}{k_1, \ldots, k_m} \prod_{j=1}^{m} \frac{\mathrm{d}^{k_j} f_j}{\mathrm{d}x^{k_j}} .$$

Finally, a function that is very useful in the study of combinatorics is the elementary symmetric function [83]

$$\sigma_{\ell,j}(y_1, \ldots, y_\ell) \triangleq \sum_{1 \le i_1 \le \cdots \le i_j \le \ell} y_{i_1} \cdots y_{i_j} ,$$

or, equivalently, if $\boldsymbol{Y} = \{y_1, \ldots, y_\ell\}$ such that $|\boldsymbol{Y}| = \ell$, [70]

$$\sigma_{\ell,j}(\boldsymbol{Y}) \triangleq \sum_{\substack{\boldsymbol{W} \subseteq \boldsymbol{Y} \\ |\boldsymbol{W}| = j}} \left(\prod_{\boldsymbol{w} \in \boldsymbol{W}} \boldsymbol{W} \right).$$

For details on computing the elementary symmetric function, see [83].

4.5.1. Problem Construction. Let there be N independent RFS multitarget states to be estimated at time k, given by

$$\begin{aligned} \boldsymbol{X}_k^{(1)} &= \{\boldsymbol{x}_{k,1}^{(1)}, \ldots, \boldsymbol{x}_{k,M^{(1)}}^{(1)}\} \\ &\vdots \\ \boldsymbol{X}_k^{(N)} &= \{\boldsymbol{x}_{k,1}^{(N)}, \ldots, \boldsymbol{x}_{k,M^{(N)}}^{(N)}\}, \end{aligned}$$

where each $\boldsymbol{x}_{k,i}^{(n)} \in \mathbb{R}^{d(n)}$. That is, each $\boldsymbol{X}_k^{(n)}$ is a finite subset of $\mathbb{R}^{d(n)}$, for some positive scalar $d(n)$ and, accordingly, the state spaces of each multitarget states are permitted to be different. Therefore, there are N sets, and any of the N types may have unique dynamics and observation profiles, the mechanizations of which are defined presently. The symbol "$\boldsymbol{x}_{k,i}^{(n)}$" can be interpreted as the i^{th} element of the n^{th} set at time index k.

Dynamical Modeling. Here, the PGFL of the transition dynamics is determined. At each reference index k, there are targets surviving from $k-1$, targets spawned from targets at $k-1$, and targets newly born into the surveillance scene at k. To model this, define the RFS of the total collection of targets at k as

$$\boldsymbol{\Xi}_{k|k-1} = \boldsymbol{\Xi}_{k|k-1}^{(1)} \cup \cdots \cup \boldsymbol{\Xi}_{k|k-1}^{(N)} \cup \boldsymbol{\Psi}_{k|k-1}^{(1)} \cup \cdots \cup \boldsymbol{\Psi}_{k|k-1}^{(N)} \cup \boldsymbol{\Gamma}_k^{(1)} \cup \cdots \cup \boldsymbol{\Gamma}_k^{(N)},$$

where $\boldsymbol{\Xi}_{k|k-1}^{(1)} \cup \cdots \cup \boldsymbol{\Xi}_{k|k-1}^{(N)}$ are the sets of surviving targets, $\boldsymbol{\Psi}_{k|k-1}^{(1)} \cup \cdots \cup \boldsymbol{\Psi}_{k|k-1}^{(N)}$ are the sets of targets spawned from the targets at $k-1$, and $\boldsymbol{\Gamma}_k^{(1)} \cup \cdots \cup \boldsymbol{\Gamma}_k^{(N)}$ are the sets of newly born targets. All of these sets are taken to be statistically independent.

The surviving sets, $\boldsymbol{\Xi}_{k|k-1}^{(n)}$, are such that

$$\boldsymbol{\Xi}_{k|k-1}^{(n)} = \boldsymbol{\Xi}_{k|k-1,1}^{(n)}(\boldsymbol{x}_{k-1,1}^{(n)}) \cup \cdots \cup \boldsymbol{\Xi}_{k|k-1,M^{(n)}}^{(n)}(\boldsymbol{x}_{k-1,M^{(n)}}^{(n)}),$$

where[13]

- $\boldsymbol{\Xi}^{(n)}_{k|k-1,i}(\boldsymbol{x}^{(n)}_{k-1,i}) = \{\emptyset\}$, i.e. target i of the n^{th} set disappears, with probability $1 - p^{(n)}_{S,k}(\boldsymbol{x}^{(n)}_{k-1,i})$ and
- $\boldsymbol{\Xi}^{(n)}_{k|k-1,i}(\boldsymbol{x}^{(n)}_{k-1,i}) = \{\boldsymbol{x}^{(n)}_{k,i}\}$, i.e. target i of the n^{th} set survives, with probability $p^{(n)}_{S,k}(\boldsymbol{x}^{(n)}_{k-1,i})$, where $\boldsymbol{x}^{(n)}_{k,i}$ is a temporally evolved state due to the Markov transition kernel $f^{(n)}(\boldsymbol{x}^{(n)}_{k,i}|\boldsymbol{x}^{(n)}_{k-1,i})$.

It can thus be seen that survival happens according to independent Bernoulli trials, and, accordingly, each of the surviving sets have PGFLs of the form

$$G^{(n)}_{\boldsymbol{\Xi}_{k|k-1,i}}[h] = 1 - p^{(n)}_{S,k}(\boldsymbol{x}^{(n)}_{k-1,i}) + p^{(n)}_{S,k}(\boldsymbol{x}^{(n)}_{k-1,i}) \int h \cdot f^{(n)}(\boldsymbol{x}^{(n)}_{k,i}|\boldsymbol{x}^{(n)}_{k-1,i}) \mathrm{d}\boldsymbol{x}^{(n)}_{k-1,i} \,. \tag{4.15}$$

Proof is given in Appendix C.8.

The spawned sets, $\boldsymbol{\Psi}^{(n)}_{k|k-1}$, are such that

$$\boldsymbol{\Psi}^{(n)}_{k|k-1} = \boldsymbol{\Psi}^{(n)}_{k|k-1,1}(\boldsymbol{x}^{(n)}_{k-1,1}) \cup \cdots \cup \boldsymbol{\Psi}^{(n)}_{k|k-1,M^{(n)}}(\boldsymbol{x}^{(n)}_{k-1,M^{(n)}}) \,.$$

Furthermore, a set spawned from target i belonging to the n^{th} set has the multitarget distribution $\beta^{(n)}(\boldsymbol{W}^{(n)}|\boldsymbol{x}^{(n)}_{k-1,i})$, where $\boldsymbol{W}^{(n)}$ is the set of spawned targets, and accordingly has the PGFL

$$G^{(n)}_{\beta}[h|\boldsymbol{x}^{(n)}_{k-1,i}] = \int h^{\boldsymbol{W}^{(n)}} \beta^{(n)}(\boldsymbol{W}^{(n)}|\boldsymbol{x}^{(n)}_{k-1,i}) \delta\boldsymbol{W}^{(n)} \,.$$

Finally, the newly born sets of type n, $\boldsymbol{\Gamma}^{(n)}_{k}$, are such that they have multitarget distribution $b(\boldsymbol{W}^{(n)}_{k})$, where $\boldsymbol{W}^{(n)}_{k}$ represents some set of newly born targets of the n^{th} type and has the PGFL

$$G^{(n)}_{b}[h] = \int h^{\boldsymbol{W}^{(n)}_{k}} b(\boldsymbol{W}^{(n)}_{k}) \delta\boldsymbol{W}^{(n)} \,.$$

[13]Here, $p^{(n)}_{S,k}(\boldsymbol{x}^{(n)}_{k-1,i})$ is the probability that a given target survived at time k given it had state $\boldsymbol{x}^{(n)}_{k-1,i}$ at $k-1$.

Therefore, defining the collection

$$\bar{\boldsymbol{\Xi}}^{(n)}_{k|k-1} = \boldsymbol{\Xi}^{(n)}_{k|k-1} \cup \boldsymbol{\Psi}^{(n)}_{k|k-1} \cup \boldsymbol{\Gamma}^{(n)}_{k} \,,$$

i.e. grouping all surviving, spawned, and born targets by type, n, means the PGFL of $\bar{\boldsymbol{\Xi}}^{(n)}_{k|k-1}$ (i.e. the PGFL of the n^{th} set's law of motion) can be written as

$$\begin{aligned} G^{(n)}_{k|k-1}[h|\boldsymbol{X}^{(n)}_{k-1}] &= G^{(n)}_{b}[h] \prod_{\boldsymbol{x}\in\boldsymbol{X}^{(n)}_{k-1}} \left(1 - p^{(n)}_{S,k}(\boldsymbol{x}) + p^{(n)}_{S,k}(\boldsymbol{x}) f^{(n)}_{S,k|k-1}[h]\right) G^{(n)}_{\beta}[h|\boldsymbol{x}] \\ &\overset{\text{abbr.}}{=} \underbrace{\left(q^{(n)}_{S,k} + p^{(n)}_{S,k} f^{(n)}_{S,k|k-1}[h]\right)^{\boldsymbol{X}^{(n)}_{k-1}}}_{\text{surviving/not surviving}} \underbrace{\left(G^{(n)}_{\beta}[h|\cdot]\right)^{\boldsymbol{X}^{(n)}_{k-1}}}_{\text{spawned}} \underbrace{G^{(n)}_{b}[h]}_{\text{born}} \,, \end{aligned} \tag{4.16}$$

where

$$f^{(n)}_{S,k|k-1}[h] = \int h \cdot f^{(n)}(\boldsymbol{x}^{(n)}_{k,i}|\boldsymbol{x}^{(n)}_{k-1,i}) \mathrm{d}\boldsymbol{x}^{(n)}_{k-1,i}$$

and $q_{S,k}(\boldsymbol{x}^{(n)}) \triangleq 1 - p^{(n)}_{S,k}(\boldsymbol{x}^{(n)})$.

In the manner of Mahler's two-argument PGFL in the derivation of the classical PHD filter in [87], define the N-argument PGFL (herein called a joint PGFL) for the collection of all sets at k and denote it $G_{k|k-1}[h_1, \ldots, h_N]$. That is, $G_{k|k-1}[h_1, \ldots, h_N]$ is the joint PGFL corresponding to multitarget transition kernel $f(\ldots|\ldots)$ (i.e. is the PGFL governing the dynamics of the entire system), where some $h_n \in [0, 1]$ is a scalar test function for the n^{th} set. This is the PGFL that governs the dynamics of the entire system. Since each of the N sets evolve, spawn, and birth independently, the PGFL of the set transitions is given as

$$G_{k|k-1}[h_1, \ldots, h_N] = G^{(1)}_{k|k-1}[h_1|\boldsymbol{X}^{(1)}_{k-1}] \cdots G^{(N)}_{k|k-1}[h_N|\boldsymbol{X}^{(N)}_{k-1}] \,. \tag{4.17}$$

Equation 4.17 will be informally referred to as the "transition PGFL" for convenience. Note that this is not the PGFL corresponding to the prior joint density $\pi(\boldsymbol{X}^{(1)}_{k}, \ldots, \boldsymbol{X}^{(N)}_{k}|\boldsymbol{Z}_{1:k-1})$ (which will be denoted $G_{k|k-1}[h_1, \ldots, h_N|\boldsymbol{Z}_{1:k-1}]$ – take note of the conditioning); instead, this is the PGFL of the joint multitarget transition density $f(\ldots|\ldots)$.

Observation Modeling. Here, the PGFL of the observation process is determined. At each index k, a set of cluttered sensor returns are collected according to the random observation set

$$\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_k^{(1)} \cup \cdots \cup \boldsymbol{\Sigma}_k^{(N)} \cup \boldsymbol{\Theta}_k \,,$$

where $\boldsymbol{\Sigma}_k^{(n)}$ is the random observation set corresponding to the n^{th} multitarget state and $\boldsymbol{\Theta}_k$ is a Poisson clutter process.[14] In particular, $\boldsymbol{\Sigma}_k^{(n)}$ is such that

$$\boldsymbol{\Sigma}_k^{(n)} = \boldsymbol{\Sigma}_{k,1}^{(n)} \cup \cdots \cup \boldsymbol{\Sigma}_{k,M^{(n)}}^{(n)} \,,$$

where $\boldsymbol{\Sigma}_{k,i}^{(n)}$ is the observation set produced by state $\boldsymbol{x}_{k,i}^{(n)}$ such that[15]

- $\boldsymbol{\Sigma}_{k,i}^{(n)} = \{\emptyset\}$, i.e. a missed detection occurs, with probability $1 - p_{D,k}^{(n)}(\boldsymbol{x}_{k,i}^{(n)})$ and
- $\boldsymbol{\Sigma}_{k,i}^{(n)} = \{\boldsymbol{z}(\boldsymbol{x}_{k,i}^{(n)})\}$, i.e. a detection occurs, with probability $p_{D,k}^{(n)}(\boldsymbol{x}_{k,i}^{(n)})$, where $\boldsymbol{z}(\boldsymbol{x}_{k,i}^{(n)})$ is a random vector-valued observation with corresponding (single-target) likelihood $g^{(n)}(\boldsymbol{z}|\boldsymbol{x}_{k,i}^{(n)})$.

That is, measurements are produced according to independent Bernoulli trials.

As was done in the dynamics modeling, define a joint PGFL, this time with $N+1$ arguments to capture the influence of collected measurements, as

$$F[g, h_1, \ldots, h_N] \triangleq \int \cdots \int h_1^{\boldsymbol{X}_k^{(1)}} \cdots h_N^{\boldsymbol{X}_k^{(N)}} \times G[g|\boldsymbol{X}_k^{(1)}, \ldots, \boldsymbol{X}_k^{(N)}] \pi(\boldsymbol{X}_k^{(1)}, \ldots, \boldsymbol{X}_k^{(N)}|\boldsymbol{Z}_{1:k-1}) \delta\boldsymbol{X}_k^{(1)} \cdots \delta\boldsymbol{X}_k^{(N)} \,,$$

[14] Certainly, this process need not be restricted to being a Poisson process, but here it is modeled as such due to the immense utility and ease of use this model offers.

[15] The term $p_{D,k}^{(n)}(\boldsymbol{x}_{k,i}^{(n)})$ is the probability that a target is detected given it has state $\boldsymbol{x}_{k,i}^{(n)}$.

where[16]

$$G[g|\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)}] \triangleq \int g^{\boldsymbol{Z}_k} g(\boldsymbol{Z}_k|\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)})\delta\boldsymbol{Z}_k$$

with k^{th} collected observation set $\boldsymbol{Z}_k = \{\boldsymbol{z}_{1,k},\ldots,\boldsymbol{z}_{m,k}\} \subset \boldsymbol{\Sigma}_k$, multitarget likelihood function $g(\cdot|\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)})$, prior joint multitarget density $\pi(\cdot|\boldsymbol{Z}_{1:k-1})$, and set of all measurements up to and including $\boldsymbol{Z}_k$ denoted as $\boldsymbol{Z}_{1:k}$. Note that $G[g|\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)}]$ is the PGFL of $g(\cdot|\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)})$ and is called the observation PGFL. To compress these expressions, the abbreviations

$$\begin{aligned}
F[g,h_1,\ldots,h_N] &\overset{\text{abbr.}}{=} F[g,h_{1:N}] \\
h_1^{\boldsymbol{X}_k^{(1)}}\cdots h_N^{\boldsymbol{X}_k^{(N)}} &\overset{\text{abbr.}}{=} h^{\boldsymbol{X}_k^{(1:N)}} \\
G[g|\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)}] &\overset{\text{abbr.}}{=} G[g|\boldsymbol{X}_k^{(1:N)}] \\
\pi(\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)}|\boldsymbol{Z}_{1:k-1}) &\overset{\text{abbr.}}{=} \pi(\boldsymbol{X}_k^{(1:N)}|\boldsymbol{Z}_{1:k-1}) \\
\delta\boldsymbol{X}_k^{(1)}\cdots\delta\boldsymbol{X}_k^{(N)} &\overset{\text{abbr.}}{=} \delta\boldsymbol{X}_k^{(1:N)}
\end{aligned}$$

are employed in the following, and these abbreviations yield

$$F[g,h_{1:N}] = \int h^{\boldsymbol{X}_k^{(1:N)}} G[g|\boldsymbol{X}_k^{(1:N)}]\pi(\boldsymbol{X}_k^{(1:N)}|\boldsymbol{Z}_{1:k-1})\delta\boldsymbol{X}_k^{(1:N)} \tag{4.18}$$

$$G[g|\boldsymbol{X}_k^{(1:N)}] = \int g^{\boldsymbol{Z}_k} g(\boldsymbol{Z}_k|\boldsymbol{X}_k^{(1:N)})\delta\boldsymbol{Z}_k \tag{4.19}$$

as more compact expressions for the joint and observation PGFLs, respectively.

Since the clutter process $\boldsymbol{\Theta}_k$ is Poisson, it has PGFL [83]

$$G_{\boldsymbol{\Theta}}[g] = \exp\{\lambda c[g] - \lambda\} \tag{4.20}$$

[16]Note that the test function g and the multitarget likelihood $g(\boldsymbol{Z}_k|\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)})$ are both denoted by "g". This is a possible source of confusion, but is adopted for consistency between sections of this dissertation and due to convention; Practically speaking, there will be no risk of confusion when one looks at the function arguments.

with clutter intensity λ and $c[g] \triangleq \int gc(\boldsymbol{z})\mathrm{d}z$ where $\boldsymbol{c}(\boldsymbol{z})$ is the spatial density of the clutter. Additionally, the PGFL of the i^{th} observation of the n^{th} set, $\boldsymbol{\Sigma}_{k,i}^{(n)}$, is given as

$$G_{\boldsymbol{\Sigma}_{k,i}}^{(n)}[g] = 1 - p_{D,k}^{(n)}(\boldsymbol{x}_{k,i}^{(n)}) + p_{D,k}^{(n)}(\boldsymbol{x}_{k,i}^{(n)}) f_{D,k}^{(n)}[g]\,.$$

Proof follows in the same way as given in Appendix C.8.

Since observations are taken to be independent, it can then be shown that the n^{th} observation set's PGFL is given as

$$\begin{aligned} G[g|\boldsymbol{X}_k^{(n)}] &= \prod_{\boldsymbol{x}\in\boldsymbol{X}_k^{(n)}} \left(1 - p_{D,k}^{(n)}(\boldsymbol{x}) + p_{D,k}^{(n)}(\boldsymbol{x}) f_{D,k}^{(n)}[g]\right) \\ &\overset{\text{abbr.}}{=} \left(1 - p_{D,k}^{(n)} + p_{D,k}^{(n)} f_{D,k}^{(n)}[g]\right)^{\boldsymbol{X}_k^{(n)}} \end{aligned} \tag{4.21}$$

where

$$f_{D,k}^{(n)}[g] = \int g \cdot g^{(n)}(\boldsymbol{z}|\boldsymbol{x}^{(n)})\mathrm{d}\boldsymbol{z}$$

and $q_{D,k}^{(n)}(\boldsymbol{x}^{(n)}) \triangleq 1 - p_{D,k}^{(n)}(\boldsymbol{x}^{(n)})$ is the probability of missed detection. To see this, recall that the PGFL of a union of independent observation processes is the product of each process's PGFL [83], and that the PGFL of each $\boldsymbol{\Sigma}_{k,i}^{(n)} \subset \boldsymbol{\Sigma}_k^{(n)}$ is $q_{D,k}^{(n)}(\boldsymbol{x}_{k,i}^{(n)}) + p_{D,k}^{(n)}(\boldsymbol{x}_{k,i}^{(n)}) f_{D,k}^{(n)}[g]$. Then, as an extension of this fact, assuming that each of the N multitarget states generate observations independently, it can be shown that the PGFL of the observation RFS $\boldsymbol{\Sigma}_k$ is

$$G[g|\boldsymbol{X}_k^{(1:N)}] = \underbrace{\exp\{\lambda c[g] - \lambda\}}_{\text{clutter}} \underbrace{\prod_{n=1}^{N} \left(q_{D,k}^{(n)} + p_{D,k}^{(n)} f_{D,k}^{(n)}[g]\right)^{\boldsymbol{X}_k^{(n)}}}_{\text{observed/not observed}}. \tag{4.22}$$

The Desired Recursion and PGFLs. As was described in Section 4.1, the multitarget Bayes filter is formed using recursive applications of the multitarget analogs of the Chapman-Kolmogorov equation and Bayes' rule. For a filter to estimate multiple sets simultaneously, the same principles are applied but to the joint multitarget density of all N sets. This is extended here to PHDs, producing a PHD filter that simultaneously estimates

the PHDs of the N sets. As with Mahler's PHD filter derivation in [87], this procedure relies on PGFLs, or, more specifically, derivatives of those PGFLs. Using Eqs. (4.13) and (4.14), the derivative relationship between joint PGFL and PHD-product for the Chapman-Kolmogorov equation, producing the prior PHD-product, is[17]

$$\prod_{n=1}^{N} v_k^{-(n)}(\boldsymbol{x}_k^{(n)}) = \frac{\delta^N G_{k|k-1}}{\delta \boldsymbol{x}_k^{(1)} \dots \delta \boldsymbol{x}_k^{(N)}}[h_1, \dots, h_N | \boldsymbol{Z}_{1:k-1}]\Big|_{h_1=\dots=h_N=1},$$

and, similarly, it can be shown that the joint PGFL of Bayes' rule is given as

$$G_{k|k}[h_1, \dots, h_N | \boldsymbol{Z}_{1:k}] = \frac{\frac{\delta^m F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m}[0, h_1, \dots, h_N]}{\frac{\delta^m F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m}[0, 1, \dots, 1]} \tag{4.23}$$

and thus the posterior PHD-product is

$$\prod_{n=1}^{N} v_k^{+(n)}(\boldsymbol{x}_k^{(n)}) = \frac{\delta^N G_{k|k}}{\delta \boldsymbol{x}_k^{(1)} \cdots \delta \boldsymbol{x}_k^{(N)}}[h_1, \dots, h_N | \boldsymbol{Z}_{1:k}]\Big|_{h_1=\dots=h_N=1}. \tag{4.24}$$

Proof of the joint PGFL for Bayes' rule, Eq. (4.23), can be found in Appendix C.11.

4.5.2. Predictor. Let $v_k^{-(n)}(\boldsymbol{x}^{(n)})$ denote the *a priori* intensity of the n^{th} set. It can be shown that the predicted PHDs for each of the N targets are given by, $\forall\ n = \{1, \dots, N\}$,

$$\begin{aligned} v_k^{-(n)}(\boldsymbol{x}^{(n)}) = &\int p_{S,k}^{(n)}(\boldsymbol{x}_k^{(n)}) f^{(n)}(\boldsymbol{x}_k^{(n)} | \boldsymbol{x}_{k-1}^{(n)}) v_{k-1}^{+(n)}(\boldsymbol{x}_{k-1}^{(n)}) \mathrm{d}\boldsymbol{x}_{k-1}^{(n)} \\ &+ \int \beta^{(n)}(\boldsymbol{x}_k^{(n)} | \boldsymbol{x}_{k-1}^{(n)}) v_{k-1}^{+(n)}(\boldsymbol{x}_{k-1}^{(n)}) \mathrm{d}\boldsymbol{x}_{k-1}^{(n)} + \gamma_k^{(n)}(\boldsymbol{x}_k^{(n)}), \end{aligned} \tag{4.25}$$

where $v_{k-1}^{+(n)}(\boldsymbol{x}_{k-1}^{(n)})$ is the posterior PHD of the n^{th} set at the previous time step, $p_{S,k}^{(n)}(\boldsymbol{x}_k^{(n)})$ is the probability of survival, $f^{(n)}(\boldsymbol{x}_k^{(n)} | \boldsymbol{x}_{k-1}^{(n)})$ is the standard single-target transition density, $\beta^{(n)}(\boldsymbol{x}_k^{(n)} | \boldsymbol{x}_{k-1}^{(n)})$ is the spawning intensity, and $\gamma_k^{(n)}(\cdot)$ is the intensity of the birth RFS $\boldsymbol{\Gamma}_k^{(n)}$. Inspection of this result yields anticipated conclusions: the prior PHDs of each of the N sets

[17] Here, the substitution bar $(\cdot)|_{h_1=\dots=h_N=1}$ is used to emphasize that the h_i are only set equal to 1 *after* the derivatives have been taken.

contain a term due to surviving targets, spawned targets, and newly born targets, as the first, second, and third terms of Eq. (4.25), respectively. Note that under this construction, each of the N sets can have their own distinct dynamics models and properties.

Proof is given in Appendix C.9.

Remark 4.6. *Note that Eq. (4.25) is simply N standard PHD filter predictions, as described in [87], or, if consider parameters are employed, can be N consider PHD filter predictions as described in Section 4.2.1. This is an intuitive result, as independent target motion of one set should be unaffected by the presence of another. Therefore, as with the PHD filter, this multiple-set PHD filter prediction is exact, and this result is not subject to any Poisson approximation. A Poisson approximation is only required to produce a usable corrector stage.*

4.5.3. Corrector. Let the prior and posterior PHD-products be written with the shorthand notation

$$\prod_{n=1}^{N} v_k^{-(n)}(\boldsymbol{x}_k^{(n)}) \overset{\text{abbr.}}{=} \Pi v_k^- \qquad \text{and} \qquad \prod_{n=1}^{N} v_k^{+(n)}(\boldsymbol{x}_k^{(n)}) \overset{\text{abbr.}}{=} \Pi v_k^+ ,$$

respectively, and assume that each of the sets' prior density is (approximately) Poisson. Then, given a collected observation of the N sets, $\boldsymbol{Z}_k$, the posterior PHD-product is given as

$$\Pi v_k^+ = \left[q_{D,k}^{\boldsymbol{N}} + \sum_{\boldsymbol{W} \subseteq_\emptyset \boldsymbol{N}} q_{D,k}^{\boldsymbol{N} \setminus \boldsymbol{W}} p_{D,k}^{\boldsymbol{W}} \sum_{\substack{\boldsymbol{Y} \subseteq_\emptyset \boldsymbol{Z}_k \\ |\boldsymbol{Y}| = |\boldsymbol{W}|}} \sigma_{\ell,j} \left(\frac{g^{(w_1)}(\boldsymbol{y}_1 | \boldsymbol{x}^{(w_1)})}{a(\boldsymbol{y}_1)}, \ldots, \frac{g^{(w_\ell)}(\boldsymbol{y}_\ell | \boldsymbol{x}^{(w_\ell)})}{a(\boldsymbol{y}_\ell)} \right) \right] \Pi v_k^- , \tag{4.26}$$

where each $w_i \in \boldsymbol{W}$, a sum over $\boldsymbol{A} \subseteq_\emptyset \boldsymbol{B}$ denotes a sum over all nonempty subsets of $\boldsymbol{B}$, $\sigma_{\ell,j}(\cdot)$ denotes the elementary symmetric function of degree j in ℓ variables, $g^{(n)}(\boldsymbol{z}|\boldsymbol{x}^{(n)})$ is the single target likelihood corresponding to the n^{th} set, and

$$\boldsymbol{N} = \{1, \ldots, N\}$$

$$j = |\boldsymbol{W}|$$

$$\ell = |\boldsymbol{Y}|$$
$$a(\boldsymbol{z}) = \kappa_k(\boldsymbol{z}) + \sum_{n=1}^{N} v_k^{-(n)}\left[p_{D,k}^{(n)}(\boldsymbol{x}_k^{(n)})g^{(n)}(\boldsymbol{z}|\boldsymbol{x}_k^{(n)})\right]$$
$$v_k^{-(n)}[h] = \int h \cdot v_k^{-(n)}(\boldsymbol{x}_k^{(n)})\mathrm{d}\boldsymbol{x}^{(n)}\,.$$

Here, the aforementioned product notation is generalized a bit to imply

$$q_{D,k}^{\boldsymbol{N}} = \prod_{n\in\boldsymbol{N}} q_{D,k}^{(n)}(\boldsymbol{x}^{(n)})$$

and similarly for $p_{D,k}^{\boldsymbol{N}}$, i.e. it produces a product of detection/missed detection probabilities for all of the sets in $\boldsymbol{N}$.

Proof of this result is given in Appendix C.10.

If, instead, one seeks to individually compute any or all of the N set's intensities directly, $v_k^{+(n)}(\cdot)$ can be computed as

$$v_k^{+(n)}(\boldsymbol{x}_k^{(n)}) = \left(\prod_{i\neq n}^{N}\hat{N}_k^{+(i)}\right)^{-1}\left[q_{D,k}(\boldsymbol{x}_k^{(n)})\prod_{i\neq n}^{N}v_k^{-(i)}[q_{D,k}(\boldsymbol{x}_k^{(i)})] + \sum_{\boldsymbol{W}\subseteq_\emptyset\boldsymbol{N}}\left(\prod_{i\in\boldsymbol{N}\setminus\boldsymbol{W}}v_k^{-(i)}[q_{D,k}(\boldsymbol{x}_k^{(i)})]\right)\times\sum_{\substack{\boldsymbol{Y}\subseteq_\emptyset\boldsymbol{Z}_k\\|\boldsymbol{Y}|=|\boldsymbol{W}|}}\sigma_{\ell,j}\left(\phi(\boldsymbol{y}_1),\ldots,\frac{g^{(n)}(\boldsymbol{y}_{i_p}|\boldsymbol{x}^{(n)})}{a(\boldsymbol{y}_{i_p})},\ldots,\phi(\boldsymbol{y}_\ell)\right)\right]v_k^{-(n)}(\boldsymbol{x}_k^{(n)})\,, \tag{4.27}$$

where

$$\phi(\boldsymbol{y}_i) = \frac{v_k^{-(w_i)}[p_{D,k}^{(w_i)}(\boldsymbol{x}_k^{(w_i)})g^{(w_i)}(\boldsymbol{y}_i|\boldsymbol{x}_k^{(w_i)})]}{a(\boldsymbol{y}_i)}\,,$$

$\hat{N}_k^{+(i)}$ denotes the posterior cardinality estimate of the i^{th} set, and the index i_p is such that $n = w_{i_p} \in \boldsymbol{W}$. This expression is obtained by marginalizing Eq. (4.26) over all $\boldsymbol{x}_k^{(i)}$ such that $i \neq n$, and this result serves as a more practical avenue to implementing a multiple

set PHD filter. However, the interpretation remains the same: within the square brackets, the first term describes the event that all targets failed to be detected and the second term accounts for all association possibilities between the N sets.

Proof is given in Appendix C.12.

An interesting distinction between this result and conventional PHD filtering methods appears: computing the n^{th} *a posteriori* PHD requires the posterior cardinality estimates, $\hat{N}_k^{+(i)}$ of the other $(N-1)$ PHDs. That is, even if one seeks to compute the posterior PHD of only one set, the posterior cardinality estimates of all of the remaining sets must be computed as well, appearing as the divisor $\prod_{i\neq n}^{N} \hat{N}_k^{+(i)}$. Interestingly, but perhaps unsurprisingly, this only requires the posterior cardinality estimates and not any measurement-updated spatial information.[18] Further, the posterior PHD of the n^{th} set is only affected in *scale* by the cardinality estimates of the other sets. This is to say that only the cardinality estimation of the n^{th} set is affected by the other sets and that, by contrast, spatial estimation of the posterior PHDs is independent of the other PHDs. To see why this is the case, observe that the integral of $\prod_{n=1}^{N} v_k^{+(n)}(\cdot)$ over all N state spaces is

$$\begin{aligned}\int \cdots \int \left(\prod_{n=1}^{N} v_k^{+(n)}(\boldsymbol{x}_k^{(n)}) \right) \mathrm{d}\boldsymbol{x}_k^{(1)} \cdots \mathrm{d}\boldsymbol{x}_k^{(N)} &= \prod_{n=1}^{N} \int v_k^{+(n)}(\boldsymbol{x}_k^{(n)}) \mathrm{d}\boldsymbol{x}_k^{(n)} \\ &= \prod_{n=1}^{N} \hat{N}_k^{+(n)} .\end{aligned}$$

That is, this integral over all N state spaces does not yield, as one may expect, the total number of targets within all N sets. Instead, since the PHD-product is estimated, it is the *product* of the number of targets in each of the N sets that is obtained. This, in some sense, adds a degree of freedom (and frustration) to determining the cardinality estimate of the n^{th} set, denoted as $\hat{N}_k^{+(n)}$. Therefore, an additional step, a cardinality estimation step, is applied.

Denoting $\rho(i_1, \ldots, i_N)$ as the joint cardinality distribution, i.e. $\rho(i_1, \ldots, i_N)$ is a proper probability mass function where i_n is an input and corresponds to a candidate cardinality of the n^{th} set, the joint cardinality distribution of the posterior density is given

[18] Why is this unsurprising? When RFSs are characterized as Poisson, as they are here, their cardinality estimates depend on only one parameter, and that parameter does not depend on that RFS's spatial density.

by

$$\rho(i_1,\ldots,i_N) = \left(\frac{\kappa_k(\cdot)}{a(\cdot)}\right)^{\boldsymbol{Z}_k} \left[\prod_{n=1}^{N} \frac{\left(\mu^{(n)} q_{D,k}^{(n)}\right)^{i_n}}{i_n!} + \sum_{\boldsymbol{W}\subseteq_{\emptyset}\boldsymbol{N}} \left(\prod_{n'\in\boldsymbol{N}\setminus\boldsymbol{W}} \frac{\left(\mu^{(n')} q_{D,k}^{(n')}\right)^{i_{n'}}}{i_{n'}!}\right) \times \left(\sum_{\substack{\boldsymbol{Y}_1\uplus\cdots\uplus\boldsymbol{Y}_j\subseteq\boldsymbol{Z}_k \\ |\boldsymbol{Y}_1|=i_{w_1},\ldots,|\boldsymbol{Y}_j|=i_{w_j}}} \prod_{n=1}^{j} \left(\frac{v_k^{-(w_n)}\left[p_{D,k}^{(w_n)} g^{(w_n)}(\cdot|\boldsymbol{x}_k^{(w_n)})\right]}{\kappa_k(\cdot)}\right)^{\boldsymbol{Y}_n}\right)\right], \tag{4.28}$$

where

$$\mu^{(n)} = \int v_k^{-(n)}(\boldsymbol{x}_k^{(n)})\mathrm{d}\boldsymbol{x}_k^{(n)}\,,$$

$\boldsymbol{A}\uplus\boldsymbol{B}$ denotes the disjoint union (that is, if $\boldsymbol{A}\uplus\boldsymbol{B}\subseteq\boldsymbol{C}$, then $\boldsymbol{A}\cap\boldsymbol{B}=\emptyset$), and $j=|\boldsymbol{W}|$. Note that, while not explicitly denoted, the probability of detection is permitted to be state-dependent.

Proof is given in Appendix C.13.

Effectively, the sum over $\boldsymbol{Y}_1\uplus\cdots\uplus\boldsymbol{Y}_j\subseteq\boldsymbol{Z}_k$ such that $|\boldsymbol{Y}_1|=i_{w_1},\ldots,|\boldsymbol{Y}_j|=i_{w_j}$ is the sum over all disjoint subsets $\boldsymbol{Y}_n\subset\boldsymbol{Z}_k$, each having exactly i_n elements. This divides $\boldsymbol{Z}_k$ into assignment possibilities between each of the N sets, the $\boldsymbol{W}$ account for detection/missed detection possibilities between each of the N sets, and $|\boldsymbol{Y}_n|=i_{w_n}$ implies that the set corresponding to $w_n\in\boldsymbol{W}$ is assigned i_{w_n} detections. It might be noted that this sum appears reminiscent of the elementary symmetric function, but its use here would require some generalization. This expression's relationship to the standard elementary symmetric function will become more clear when the marginal cardinality distribution is presented.

Given this expression, cardinality estimates are then obtained by finding

$$\hat{N}_k^{+(1)},\ldots,\hat{N}_k^{+(N)} = \underset{i_1,\ldots,i_N}{\arg\max}\,\rho(i_1,\ldots,i_N)\,. \tag{4.29}$$

Remark 4.7 (An Example)**.** *As a specific example, take* $N = 3$*, and therefore* $\boldsymbol{N} = \{1, 2, 3\}$*. Let the collected measurement be such that* $m = 4$*; that is,* $\boldsymbol{Z}_k = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3, \boldsymbol{z}_4\}$*. One possible subset of* $\boldsymbol{N}$ *is* $\boldsymbol{W} = \{1, 2\}$*, and for this* $\boldsymbol{W}$*, all possible subsets* $\boldsymbol{Y}_i$ *of* $\boldsymbol{Z}_k$ *are*

$$\begin{array}{ccccccc} \{\emptyset\} & \{\boldsymbol{z}_1\} & \{\boldsymbol{z}_2\} & \{\boldsymbol{z}_3\} & \{\boldsymbol{z}_4\} & & \\ \{\boldsymbol{z}_1, \boldsymbol{z}_2\} & \{\boldsymbol{z}_1, \boldsymbol{z}_3\} & \{\boldsymbol{z}_1, \boldsymbol{z}_4\} & \{\boldsymbol{z}_2, \boldsymbol{z}_3\} & \{\boldsymbol{z}_2, \boldsymbol{z}_4\} & \{\boldsymbol{z}_3, \boldsymbol{z}_4\} & . \\ \{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3\} & \{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_4\} & \{\boldsymbol{z}_1, \boldsymbol{z}_3, \boldsymbol{z}_4\} & \{\boldsymbol{z}_2, \boldsymbol{z}_3, \boldsymbol{z}_4\} & \{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3, \boldsymbol{z}_4\} & & \end{array}$$

Selecting some $i_{w_1} = i_1 = 1$ *and* $i_{w_2} = i_2 = 2$ *means that the disjoint sum is taken over all* $\boldsymbol{Y}_n$ *such that*

$$\begin{aligned} \boldsymbol{Y}_1 &= \{\boldsymbol{z}_1\},\ \{\boldsymbol{z}_2\},\ \{\boldsymbol{z}_3\},\ \{\boldsymbol{z}_4\} \\ \boldsymbol{Y}_2 &= \{\boldsymbol{z}_1, \boldsymbol{z}_2\},\ \{\boldsymbol{z}_1, \boldsymbol{z}_3\},\ \{\boldsymbol{z}_1, \boldsymbol{z}_4\},\ \{\boldsymbol{z}_2, \boldsymbol{z}_3\},\ \{\boldsymbol{z}_2, \boldsymbol{z}_4\},\ \{\boldsymbol{z}_3, \boldsymbol{z}_4\}. \end{aligned}$$

This ensures that, under a given assignment hypothesis that $|\boldsymbol{X}_k^{(n)}| = i_n$*, only* i_n *measurements in* $\boldsymbol{Z}_k$ *are assigned to the* n^{th} *set. Loosely speaking,* i_n *is a running variable in the joint cardinality distribution used to ask "What is the probability that the cardinality of set* n *is* i_n*?" Of course, one must select an* $i_n \ \forall\ n \in \boldsymbol{N}$ *to evaluate this function. Or, one could use the marginals, as will be discussed next.*

As with the posterior PHDs, rather than managing the entire joint cardinality distribution, it is more practical to compute and store N marginal cardinality distributions. Each marginal is given as

$$\rho^{(n)}(i_n) = \left(1 - \frac{v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\cdot|\boldsymbol{x}^{(n)})\right]}{a(\cdot)}\right)^{\boldsymbol{Z}_k} \left[\frac{\left(\mu^{(n)} q_{D,k}^{(n)}\right)^{i_n}}{i_n!} + \sigma_{\ell,i_n}\left(\psi(\boldsymbol{z}_1), \ldots, \psi(\boldsymbol{z}_\ell)\right)\right], \tag{4.30}$$

where

$$\psi(\boldsymbol{z}) = \frac{v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\boldsymbol{z}|\boldsymbol{x}^{(n)})\right]}{a(\boldsymbol{z}) - v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\boldsymbol{z}|\boldsymbol{x}^{(n)})\right]}.$$

Now, cardinality estimation is performed by the scalar maximization

$$\hat{N}_k^{+(n)} = \arg\max_{i_n} \rho^{(n)}(i_n)\,. \tag{4.31}$$

Rather than solving the joint maximization problem over $i_1, \ldots, i_N$, N one-dimensional maximizations are performed.

Proof is given in Appendix C.14.

Inspecting this result immediately indicates its advantages: this expression is substantially simpler than that of Eq. (4.28), and, in fact, computing N of these marginal distributions is preferable to computing all of $\rho(i_1, \ldots, i_N)$. This is an unsurprising, but significant, result that arises primarily from the combinatorics inherent to this problem.

Interestingly, the sum over all disjoint subsets in Eq. (4.28) reduces to a term expressed as an elementary symmetric function, indicating that the sum over all disjoint subsets in Eq. (4.28) is, in some sense, a generalization of the elementary symmetric function. While not necessarily a useful conclusion, it does serve to lend some intuition to the problem.

Summary. In general, the prediction for the N PHDs is done using Eq. (4.25), and the corrector for the PHD-product of all N sets is performed using the PHD update of Eq. (4.26) and cardinality estimation with Eqs. (4.28) and (4.29). A practical implementation may instead prefer computing the N marginal posterior PHDs directly and separately using the update of Eq. (4.27) and corresponding marginal cardinality estimation of Eqs. (4.30) and (4.31).

Remark 4.8. *Inspection of the update equations makes it clear that the independence between sets is violated upon completion of an update, and, thus, subsequent prediction steps are approximate rather than exact (with respect to the initial PHD). It is possible to generalize this result further, but this is omitted here.*

Remark 4.9. *It may seem alarming at first that the much-adored linear complexity of the standard PHD filter has been exchanged for a generally intractable combinatorial sum. In fact, the same thing occurs in Clark's general PHD filter of [130] (and as discussed in Mahler's second textbook [101]). While at first this may cause the practical utility of such*

a filter to be brought into question, [130] notes that practical approximations have been developed for combinatorial sums of similar type in extended target tracking, such as in Swain and Clark [122] and Orguner et al. [123]. The numerical results of Section 4.5.6 aim to illustrate the utility of the derived filter and that implementations for more complex scenarios are not infeasible.

Remark 4.10 (The $N = 1$ Case)**.** *It is comforting to note that in the case of $N = 1$, Eq. (4.26) becomes the standard PHD filter, as expected. To see this, set $N = 1$ and manipulate to obtain*

$$\begin{aligned} v_k^{+(1)}(\boldsymbol{x}_k^{(1)}) &= \left[q_{D,k}^{(1)} + \sum_{\boldsymbol{W}\subseteq\emptyset\{1\}} q_{D,k}^{\{1\}\setminus\boldsymbol{W}} p_{D,k}^{\boldsymbol{W}} \sum_{\substack{\boldsymbol{Y}\subseteq_\emptyset \boldsymbol{Z}_k \\ |\boldsymbol{Y}|=|\boldsymbol{W}|}} \sigma_{\ell,j} \left(\frac{g^{(w_1)}(\boldsymbol{y}_1|\boldsymbol{x}^{(w_1)})}{a(\boldsymbol{y}_1)} \right) \right] v_k^{-(1)}(\boldsymbol{x}_k^{(1)}) \\ &= \left[q_{D,k}^{(1)}(\boldsymbol{x}_k^{(1)}) + p_{D,k}^{(1)}(\boldsymbol{x}_k^{(1)}) \sum_{\boldsymbol{z}\in\boldsymbol{Z}_k} \sigma_{\ell,j} \left(\frac{g^{(w_1)}(\boldsymbol{z}|\boldsymbol{x}^{(w_1)})}{a(\boldsymbol{z})} \right) \right] v_k^{-(1)}(\boldsymbol{x}_k^{(1)}) \\ &= \left[q_{D,k}(\boldsymbol{x}_k^{(1)}) + \sum_{\boldsymbol{z}\in\boldsymbol{Z}_k} \frac{p_{D,k}^{(1)}(\boldsymbol{x}_k^{(1)}) g^{(1)}(\boldsymbol{z}|\boldsymbol{x}^{(1)})}{\kappa_k(\boldsymbol{z}) + v_k^{-(1)}\left[p_{D,k}^{(1)}(\boldsymbol{x}_k^{(1)}) g^{(1)}(\boldsymbol{z}|\boldsymbol{x}^{(1)}) \right]} \right] v_k^{-(1)}(\boldsymbol{x}_k^{(1)}) , \end{aligned}$$

which is, of course, just the standard PHD filter corrector of Eq. (4.3).

Remark 4.11 (The $N = 2$ Case)**.** *It seems that that the case where $N = 2$, i.e. one is tracking two sets, is a particularly attractive use-case scenario for such a filter, and, therefore, this special case is described here. The predictor equations are obtained trivially from what was presented before, but the corrector becomes*

$$\begin{aligned} \Pi v_k^{+} = \Bigg[q_{D,k}^{(1)} q_{D,k}^{(2)} + \sum_{k=1}^{m} \left\{ \frac{q_{D,k}^{(2)} p_{D,k}^{(1)} g^{(1)}(\boldsymbol{z}_k|\boldsymbol{x}^{(1)}) + q_{D,k}^{(1)} p_{D,k}^{(2)} g^{(2)}(\boldsymbol{z}_k|\boldsymbol{x}^{(2)})}{a(\boldsymbol{z}_k)} \right\} \\ + p_{D,k}^{(1)} p_{D,k}^{(2)} \sum_{\substack{i,j \\ i\neq j}}^{m} \left\{ \frac{g^{(1)}(\boldsymbol{z}_i|\boldsymbol{x}^{(1)}) g^{(2)}(\boldsymbol{z}_j|\boldsymbol{x}^{(2)})}{a(\boldsymbol{z}_i) a(\boldsymbol{z}_j)} \right\} \Bigg] \Pi v_k^{-} , \end{aligned}$$

where the sum over i, j is the sum over all $\binom{n}{2}$ combinations of i, j such that $i \neq j$ with repetition (that is $(i, j) = (1, 2)$ and $(i, j) = (2, 1)$ each have their own term) and $i, j \leq m$. Explicit definition of the cardinality step for this case is omitted but his handily obtained from simplification of the presented results.

This is an interesting case, as it lends intuition to a user in the the way this filter works. The first term accounts for the case where neither of the sets are detected. The second term accounts for the case that only one of the sets, or the other, is detected. The final term accounts for all the possibilities of assigning the collected measurements to one set and the other.

4.5.4. GM Implementations. The results presented in the previous section serve as a theoretical basis for a PHD filter that simultaneously estimates multiple sets, and this section utilizes GMs to produce a practical recursion.

Predictor. Given that the modeling decisions of Section 4.5.1 are obeyed and assuming that target survival is state independent, a GM-implementation of the prediction in Eq. (4.25) is simply N standard GM PHD filter predictions [69]. Alternatively, if consider parameters are employed, this prediction becomes N GM consider PHD filter predictions as developed by this dissertation in Section 4.2.1.1.

Corrector. Let the observations of the N sets be generated according to the linear relationship $\boldsymbol{z} = \boldsymbol{H}_{x,k}^{(n)} \boldsymbol{x}_k^{(n)} + \boldsymbol{H}_{v,k}^{(n)} \boldsymbol{v}_k^{(n)}$ such that[19]

$$g^{(n)}(\boldsymbol{z}|\boldsymbol{x}_k^{(n)}) = p_g\left(\boldsymbol{z}\,;\, \boldsymbol{H}_{x,k}^{(n)} \boldsymbol{x}_k^{(n)}, \boldsymbol{H}_{v,k}^{(n)} \boldsymbol{P}_{vv,k}^{(n)} (\boldsymbol{H}_{v,k}^{(n)})^T\right) .$$

Then, given N GMs describing the PHDs of each of the estimated sets of the form

$$v_k^{-(n)}(\boldsymbol{x}_k^{(n)}) = \sum_{\ell=1}^{L_k^{-(n)}} w_{\ell,k}^{-(n)} p_g\left(\boldsymbol{x}_k^{(n)}\,;\, \boldsymbol{m}_{x,\ell,k}^{-(n)}, \boldsymbol{P}_{xx,\ell,k}^{-(n)}\right)$$

[19] Nonlinear models can be accommodated using standard linearization or quadrature techniques.

and assuming that observations are generated according to a state-independent probability of detection, the n^{th} posterior PHD, divided by some to-be-determined scaling factor $\eta^{(n)}$, is given as

$$\frac{v_k^{+(n)}(\boldsymbol{x}_k^{(n)})}{\eta^{(n)}} = q_{D,k}^{\boldsymbol{N}} v_k^{-(n)}(\boldsymbol{x}_k^{(n)}) + \sum_{\boldsymbol{W}\subseteq\emptyset\boldsymbol{N}} \sum_{\substack{\boldsymbol{Y}\subseteq\emptyset\boldsymbol{Z}_k \\ |\boldsymbol{Y}|=|\boldsymbol{W}|}} \sum_{\ell=1}^{L_k^{-(n)}} \hat{w}_{\ell,k}(\boldsymbol{Y}) p_g\left(\boldsymbol{x}_k^{(n)}\,;\, \hat{\boldsymbol{m}}_{x,\ell,k}^{(n)}(\boldsymbol{y}_{i_p}), \hat{\boldsymbol{P}}_{xx,\ell,k}^{(n)}\right), \tag{4.32}$$

where, if $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{|\boldsymbol{Y}|}\}$,

$$\begin{aligned}
\hat{w}_{\ell,k}(\boldsymbol{Y}) &= q_{D,k}^{\boldsymbol{N}\setminus\boldsymbol{W}} p_{D,k}^{\boldsymbol{N}} \cdot \sigma_{|\boldsymbol{Y}|,j}\left(\phi(\boldsymbol{y}_1), \ldots, \frac{q_{\ell,k}^{(n)}(\boldsymbol{y}_{i_p})}{a(\boldsymbol{y}_{i_p})}, \ldots, \phi(\boldsymbol{y}_{|\boldsymbol{Y}|})\right) \\
q_{\ell,k}^{(n)}(\boldsymbol{y}) &= p_g\left(\boldsymbol{y}\,;\, \boldsymbol{H}_{x,k}^{(n)} \boldsymbol{m}_{x,\ell,k}^{-(n)}, \boldsymbol{P}_{zz,\ell,k}^{-(n)}\right) \\
\boldsymbol{P}_{zz,\ell,k}^{-(n)} &= \boldsymbol{H}_{x,k}^{(n)} \boldsymbol{P}_{xx,\ell,k}^{-(n)} (\boldsymbol{H}_{x,k}^{(n)})^T + \boldsymbol{H}_{v,k}^{(n)} \boldsymbol{P}_{vv,k}^{(n)} (\boldsymbol{H}_{v,k}^{(n)})^T \\
\phi(\boldsymbol{y}_i) &= \frac{1}{a(\boldsymbol{y}_i)} \sum_{\ell=1}^{L_k^{-(w_i)}} q_{\ell,k}^{(w_i)}(\boldsymbol{y}_i) w_{\ell,k}^{-(w_i)} \\
a(\boldsymbol{y}) &= \kappa_k(\boldsymbol{y}) + \sum_{n=1}^{N} \sum_{\ell=1}^{L_k^{-(n)}} p_{D,k}^{(n)} q_{\ell,k}^{(n)}(\boldsymbol{y}) w_{\ell,k}^{-(n)},
\end{aligned}$$

and the corresponding component means and covariances are given by[20]

$$\begin{aligned}
\hat{\boldsymbol{m}}_{x,\ell,k}^{(n)}(\boldsymbol{y}_{i_p}) &= \boldsymbol{m}_{x,\ell,k}^{-(n)} + \boldsymbol{K}_{x,\ell,k}^{(n)}[\boldsymbol{y}_{i_p} - \boldsymbol{H}_{x,k}^{(n)} \boldsymbol{m}_{x,\ell,k}^{-(n)}] \\
\hat{\boldsymbol{P}}_{xx,\ell,k}^{(n)} &= \boldsymbol{P}_{xx,\ell,k}^{-(n)} - \boldsymbol{K}_{x,\ell,k}^{(n)} \boldsymbol{P}_{zz,\ell,k}^{-(n)} (\boldsymbol{K}_{x,\ell,k}^{(n)})^T \\
\boldsymbol{K}_{x,\ell,k}^{(n)} &= \boldsymbol{P}_{xx,\ell,k}^{-(n)} (\boldsymbol{H}_{x,k}^{(n)})^T (\boldsymbol{P}_{zz,\ell,k}^{-(n)})^{-1},
\end{aligned}$$

where it is noted that the mean is calculated using $\boldsymbol{y}_{i_p}$, recalling that p is such that $w_{i_p} = n$. That is, only one measurement is assigned to the n^{th} set at a time, and the n^{th} set is assigned $\boldsymbol{y}_{i_p}$ to update the mean. The scaling factor is found such that the integral of $v_k^{+(n)}(\cdot)$ is equal to the number of expected targets of the n^{th} set in order to satisfy the properties of

[20] Discussion on the various forms of the covariance update are given in Section 2.1.3.

PHDs, and it can be shown that this corresponds to

$$\eta^{(n)} = \left(\prod_{i \neq n}^{N} \hat{N}_k^{+(i)} \right)^{-1} .$$

Recall that $\hat{N}_k^{+(n)}$ is the *posterior* cardinality estimate obtained by finding

$$\hat{N}_k^{+(n)} = \max_{i_n} \rho^{(n)}(i_n) ,$$

where the cardinality distribution is given due to the employed GMs as

$$\rho^{(n)}(i_n) = \left(1 - \frac{p_{D,k}^{(n)}}{a(\cdot)} \sum_{\ell=1}^{L_k^{-(n)}} q_{\ell,k}^{(n)}(\cdot) w_{\ell,k}^{-(n)} \right)^{\boldsymbol{Z}_k} \times \left[\frac{1}{i_n!} \left(q_{D,k}^{(n)} \sum_{\ell=1}^{L_k^{-(n)}} w_{\ell,k}^{-(n)} \right)^{i_n} + \sigma_{|\boldsymbol{Z}_k|,i_n} \left(\psi(\boldsymbol{z}_1), \ldots, \psi(\boldsymbol{z}_{|\boldsymbol{Z}_k|}) \right) \right]$$

where

$$\psi(\boldsymbol{z}) = \frac{p_{D,k}^{(n)} \sum_{\ell=1}^{L_k^{-(n)}} q_{\ell,k}^{(n)}(\boldsymbol{z}) w_{\ell,k}^{-(n)}}{a(\boldsymbol{z}) - p_{D,k}^{(n)} \sum_{\ell=1}^{L_k^{-(n)}} q_{\ell,k}^{(n)}(\boldsymbol{z}) w_{\ell,k}^{-(n)}} ,$$

and the fact that

$$\mu^{(n)} = \int v_k^{-(n)}(\boldsymbol{x}_k^{(n)}) \mathrm{d}\boldsymbol{x}_k^{(n)} = \sum_{\ell=1}^{L_k^{-(n)}} w_{\ell,k}^{-(n)}$$

has been utilized.

Then, the sums in Eq. (4.32) can be computed, resulting in a larger, $L_k^{+(n)}$-component GM with weights $\tilde{w}_{\ell,k}^{(n)}$, means $\boldsymbol{m}_{x,\ell,k}^{+(n)}$, and covariances $\boldsymbol{P}_{xx,\ell,k}^{+(n)}$. By inspecting Eq. (4.32), it can be seen that these weights, means, and covariances are contributed to by the processed measurement terms—$\hat{w}_{\ell,k}(\boldsymbol{Y})$, $\hat{\boldsymbol{m}}_{x,\ell,k}^{(n)}(\boldsymbol{y}_{i_p})$, and $\hat{\boldsymbol{P}}_{xx,\ell,k}^{(n)}$—as well as missed detection terms—$q_{D,k}^{\boldsymbol{N}}$, $\boldsymbol{m}_{x,\ell,k}^{-(n)}$, and $\boldsymbol{P}_{xx,\ell,k}^{-(n)}$. After collecting all of these terms, the resulting posterior

PHD for the n^{th} set is of the form

$$v_k^{+(n)}(\boldsymbol{x}_k^{(n)}) = \left(\prod_{i\neq n}^{N} \hat{N}_k^{+(i)}\right)^{-1} \sum_{\ell=1}^{L_k^{+(n)}} \tilde{w}_{\ell,k}^{(n)} p_g\left(\boldsymbol{x}_k^{(n)}\,;\, \boldsymbol{m}_{x,\ell,k}^{+(n)}, \boldsymbol{P}_{xx,\ell,k}^{+(n)}\right),$$

and this can be rewritten in the desired, final form as

$$v_k^{+(n)}(\boldsymbol{x}_k^{(n)}) = \sum_{\ell=1}^{L_k^{+(n)}} w_{\ell,k}^{+(n)} p_g\left(\boldsymbol{x}_k^{(n)}\,;\, \boldsymbol{m}_{x,\ell,k}^{+(n)}, \boldsymbol{P}_{xx,\ell,k}^{+(n)}\right),$$

with weights

$$w_{\ell,k}^{+(n)} = \hat{N}_k^{+(n)} \frac{\tilde{w}_{\ell,k}^{(n)}}{\sum_{\ell=1}^{L_k^{+(n)}} \tilde{w}_{\ell,k}^{(n)}}. \tag{4.33}$$

This result utilizes the fact that

$$\prod_{i\neq n}^{N} \hat{N}_k^{+(i)} = \frac{\prod_{i=1}^{N} \hat{N}_k^{+(i)}}{\hat{N}_k^{+(n)}}.$$

Previously, it was seen as a negative element of Eq. (4.30) that the posterior cardinality estimates for all N sets were required to find the posterior cardinality distribution of just *one* of the N sets rather than all N. Here, however, it can be seen that by using GMs and a state-independent probability of detection, this dependence is "decoupled." This is largely a result of the fact that the product of all the cardinalities has been encoded into each of the weights of the N PHDs following the update. That is, the weights that were computed and collected, $\tilde{w}_{\ell,k}^{(n)}\ \forall\ n \in \boldsymbol{N}$, share combined information representing the product of all N cardinality estimates, i.e.

$$\left(\sum_{\ell=1}^{L_k^{+(1)}} \tilde{w}_{\ell,k}^{(1)}\right) = \cdots = \left(\sum_{\ell=1}^{L_k^{+(N)}} \tilde{w}_{\ell,k}^{(N)}\right) = \hat{N}_k^{+(1)} \cdots \hat{N}_k^{+(N)}.$$

Thus the weights of the GM representing PHD of the n^{th} set are given by Eq. (4.33), where the weights are normalized to one and then scaled to its cardinality estimate.

Proof is omitted here, but these results follow from use of Lemmas 3.1 and 3.2 on the expression presented in Section 4.5.3.

A pseudocode representation for this corrector is presented in Algorithms 5–7 to assist implementing such a filter. To aid in interpreting these algorithms, it is noted that it is common and useful to perform GM storage within a computing language, such as MATLAB, in the following manner: if the GM contains L components of dimension n, the weights are stored as an $L \times 1$ vector, the means are stored as an $n \times L$ two-dimensional array, and the covariances are stored as an $n \times n \times L$ three-dimensional array. To compact this concept, these algorithms simply use a generic placeholder "concatenate$\{\cdot\}$" to represent stacking in appropriate dimensions. It should be noted that there are many more efficient ways of constructing a code-based implementation of these techniques, via vectorization and other matrix-algebra computing techniques, but these algorithms are presented for clarity rather than performance.

In Algorithm 7, a parameter $i_{\max}^{(n)}$ is defined. This parameter is an approximation to the computation of $\rho^{(n)}(i_n)$ such that i_n need not approach infinity. Practically speaking, a user simply sets $i_{\max}^{(n)}$ to a number reasonably larger than the expected maximum cardinality of $\boldsymbol{X}_k^{(n)}$.

Algorithm 5 Preparatory Computations for Multiple-Set GM PHD Filter Corrector

function COMPUTE_TERMS($\{w_{\ell,k}^{-(n)}, \boldsymbol{m}_{x,\ell,k}^{-(n)}, \boldsymbol{P}_{xx,\ell,k}^{-(n)}\}_{\ell=1}^{L_k^{-(n)}}$, $\boldsymbol{Z}_k$)
(compute PHD update terms for $n \in \boldsymbol{N}$ and observation set $\boldsymbol{Z}_k$)
for $\boldsymbol{z} \in \boldsymbol{Z}_k$ **do**
for $n = 1, \ldots, N$ **do**
for $\ell = 1, \ldots, L_k^{-(n)}$ **do**
$\boldsymbol{P}_{zz,\ell,k}^{-(n)} = \boldsymbol{H}_{x,k}^{(n)} \boldsymbol{P}_{xx,\ell,k}^{-(n)} (\boldsymbol{H}_{x,k}^{(n)})^T + \boldsymbol{H}_{v,k}^{(n)} \boldsymbol{P}_{vv,k}^{(n)} (\boldsymbol{H}_{v,k}^{(n)})^T$
$\boldsymbol{K}_{x,\ell,k}^{(n)} = \boldsymbol{P}_{xx,\ell,k}^{-(n)} (\boldsymbol{H}_{x,k}^{(n)})^T (\boldsymbol{P}_{zz,\ell,k}^{-(n)})^{-1}$
store:
$q_{\ell,k}^{(n)}(\boldsymbol{z}) = p_g(\boldsymbol{z}\,;\, \boldsymbol{H}_{x,k}^{(n)} \boldsymbol{m}_{x,\ell,k}^{-(n)}, \boldsymbol{P}_{zz,\ell,k}^{-(n)})$
$\hat{\boldsymbol{m}}_{x,\ell,k}^{(n)}(\boldsymbol{z}) = \boldsymbol{m}_{x,\ell,k}^{-(n)} + \boldsymbol{K}_{x,\ell,k}^{(n)} [\boldsymbol{z} - \boldsymbol{H}_{x,k}^{(n)} \boldsymbol{m}_{x,\ell,k}^{-(n)}]$
$\hat{\boldsymbol{P}}_{xx,\ell,k}^{(n)} = \boldsymbol{P}_{xx,\ell,k}^{-(n)} - \boldsymbol{K}_{x,\ell,k}^{(n)} \boldsymbol{P}_{zz,\ell,k}^{-(n)} (\boldsymbol{K}_{x,\ell,k}^{(n)})^T$
store:
$a(\boldsymbol{z}) = \kappa_k(\boldsymbol{z}) + \sum_{n=1}^{N} \sum_{\ell=1}^{L_k^{-(n)}} p_{D,k}^{(n)} q_{\ell,k}^{(n)}(\boldsymbol{z}) w_{\ell,k}^{-(n)}$

Algorithm 6 PHD Update for Multiple-Set GM PHD Filter Corrector (Part I)

given $\{w_{\ell,k}^{-(n)}, \boldsymbol{m}_{x,\ell,k}^{-(n)}, \boldsymbol{P}_{xx,\ell,k}^{-(n)}\}_{\ell=1}^{L_k^{-(n)}}$ for $n \in \boldsymbol{N}$ and observation set $\boldsymbol{Z}_k$

do COMPUTE_TERMS($\{w_{\ell,k}^{-(n)}, \boldsymbol{m}_{x,\ell,k}^{-(n)}, \boldsymbol{P}_{xx,\ell,k}^{-(n)}\}_{\ell=1}^{L_k^{-(n)}}$, $\boldsymbol{Z}_k$) for $n \in \boldsymbol{N}$

for $n = 1, \ldots, N$ **do**

(missed detection terms)

for $\ell = 1, \ldots, L_k^{-(n)}$ **do**

$w_{\mathrm{md},\ell}^{(n)} = q_{D,k}^{\boldsymbol{N}} w_{\ell,k}^{-(n)}$, $\boldsymbol{m}_{\mathrm{md},\ell}^{(n)} = \boldsymbol{m}_{x,\ell,k}^{-(n)}$, and $\boldsymbol{P}_{\mathrm{md},\ell}^{(n)} = \boldsymbol{P}_{xx,\ell,k}^{-(n)}$

(update terms)

declare $\boldsymbol{w}_{\mathrm{up}}^{(n)}$, $\boldsymbol{m}_{\mathrm{up}}^{(n)}$, and $\boldsymbol{P}_{\mathrm{up}}^{(n)}$ are empty 1-D, 2-D, and 3-D arrays, respectively

for $\boldsymbol{W} \subseteq \boldsymbol{N}$ such that $\boldsymbol{W} = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{|\boldsymbol{W}|}\}$ **do**

$i_p = \{i \mid w_i = n\}$

for $\boldsymbol{Y} \subseteq \boldsymbol{Z}$ such that $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{|\boldsymbol{Y}|}\}$ and $|\boldsymbol{Y}| = |\boldsymbol{W}|$ **do**

if $n \in \boldsymbol{W}$ **then**

(if n is assigned a measurement)

for $\ell = 1, \ldots, L_k^{-(n)}$ **do**

for $i = 1, \ldots, |\boldsymbol{Y}|$ such that $i \neq i_p$ **do**

$\phi(\boldsymbol{y}_i) = \frac{1}{a(\boldsymbol{y}_i)} \sum_{\ell=1}^{L_k^{-(w_i)}} q_{\ell,k}^{(w_i)}(\boldsymbol{y}_i) w_{\ell,k}^{-(w_i)}$

$c_\ell = q_{D,k}^{\boldsymbol{N}\setminus\boldsymbol{W}} p_{D,k}^{\boldsymbol{N}} \sigma_{|\boldsymbol{Y}|,|\boldsymbol{W}|}\left(\phi(\boldsymbol{y}_1), \ldots, \frac{q_{\ell,k}^{(n)}(\boldsymbol{y}_{i_p})}{a(\boldsymbol{y}_{i_p})}, \ldots, \phi(\boldsymbol{y}_{|\boldsymbol{Y}|})\right)$

$\boldsymbol{w}_{\mathrm{up}}^{(n)} = \mathrm{concatenate}\{\boldsymbol{w}_{\mathrm{up}}^{(n)}, c_\ell \cdot w_{\ell,k}^{-(n)}\}$

$\boldsymbol{m}_{\mathrm{up}}^{(n)} = \mathrm{concatenate}\{\boldsymbol{m}_{\mathrm{up}}^{(n)}, \hat{\boldsymbol{m}}_{x,\ell,k}^{(n)}(\boldsymbol{y}_{i_p})\}$

$\boldsymbol{P}_{\mathrm{up}}^{(n)} = \mathrm{concatenate}\{\boldsymbol{P}_{\mathrm{up}}^{(n)}, \hat{\boldsymbol{P}}_{xx,\ell,k}\}$

else

(if n is not assigned a measurement)

for $i = 1, \ldots, |\boldsymbol{Y}|$ **do**

$\phi(\boldsymbol{y}_i) = \frac{1}{a(\boldsymbol{y}_i)} \sum_{\ell=1}^{L_k^{-(w_i)}} q_{\ell,k}^{(w_i)}(\boldsymbol{y}_i) w_{\ell,k}^{-(w_i)}$

$d = \sigma_{|\boldsymbol{Y}|,|\boldsymbol{W}|}\left(\phi(\boldsymbol{y}_1), \ldots, \phi(\boldsymbol{y}_{|\boldsymbol{Y}|})\right)$

for $\ell = 1, \ldots, L_k^{-(n)}$ **do**

$\boldsymbol{w}_{\mathrm{up}}^{(n)} = \mathrm{concatenate}\{\boldsymbol{w}_{\mathrm{up}}^{(n)}, d \cdot w_{\ell,k}^{-(n)}\}$

$\boldsymbol{m}_{\mathrm{up}}^{(n)} = \mathrm{concatenate}\{\boldsymbol{m}_{\mathrm{up}}^{(n)}, \boldsymbol{m}_{x,\ell,k}^{-(n)}\}$

$\boldsymbol{P}_{\mathrm{up}}^{(n)} = \mathrm{concatenate}\{\boldsymbol{P}_{\mathrm{up}}^{(n)}, \boldsymbol{P}_{xx,\ell,k}^{-(n)}\}$

(add the missed detection and detection PHDs)

$j = 0$

for $i = 1, \ldots, L_k^{-(n)}$ **do**

$\tilde{w}_{j,k}^{(n)} = w_{\mathrm{md},i}$, $\boldsymbol{m}_{x,j,k}^{+(n)} = \boldsymbol{m}_{\mathrm{md},i}^{(n)}$, and $\boldsymbol{P}_{xx,j,k}^{+(n)} = \boldsymbol{P}_{\mathrm{md},i}^{(n)}$

$j \leftarrow j + 1$

for $i = 1, \ldots, \dim\{\boldsymbol{w}_{\mathrm{up}}^{(n)}\}$ **do**

$\tilde{w}_{j,k}^{(n)} = w_{\mathrm{up},i}$, $\boldsymbol{m}_{x,j,k}^{+(n)} = \boldsymbol{m}_{\mathrm{up},i}^{(n)}$, and $\boldsymbol{P}_{xx,j,k}^{+(n)} = \boldsymbol{P}_{\mathrm{up},i}^{(n)}$

$j \leftarrow j + 1$

(see Part II)

Algorithm 7 PHD Update for Multiple-Set GM PHD Filter Corrector (Part II)

given $\{\tilde{w}_{\ell,k}^{(n)}, \boldsymbol{m}_{x,\ell,k}^{+(n)}, \boldsymbol{P}_{xx,\ell,k}^{+(n)}\}_{\ell=1}^{L_k^{+(n)}}$ for $n \in \boldsymbol{N}$ from Part I
(cardinality estimation)
for $n \in \boldsymbol{N}$ **do**
  declare a parameter $i_{\max}^{(n)}$
  for $i = 1, \ldots, i_{\max}^{(n)}$ **do**
    for $j = 1, \ldots, |\boldsymbol{Z}_k|$ **do**

$$\psi(\boldsymbol{z}_j) = \frac{p_{D,k}^{(n)} \sum_{\ell=1}^{L_k^{-(n)}} q_{\ell,k}^{(n)}(\boldsymbol{z}_j) w_{\ell,k}^{-(n)}}{a(\boldsymbol{z}_j) - p_{D,k}^{(n)} \sum_{\ell=1}^{L_k^{-(n)}} q_{\ell,k}^{(n)}(\boldsymbol{z}_j) w_{\ell,k}^{-(n)}}$$

$$c = \prod_{\boldsymbol{z} \in \boldsymbol{Z}_k} \left(1 - \frac{p_{D,k}^{(n)}}{a(\boldsymbol{z})} \sum_{\ell=1}^{L_k^{-(n)}} q_{\ell,k}^{(n)}(\boldsymbol{z}) w_{\ell,k}^{-(n)}\right)$$

$$\rho^{(n)}(i) = c \cdot \left[\frac{1}{i_n!}\left(q_{D,k}^{(n)} \sum_{\ell=1}^{L_k^{-(n)}} w_{\ell,k}^{-(n)}\right)^{i_n} + \sigma_{|\boldsymbol{Z}_k|, i_n}\left(\psi(\boldsymbol{z}_1), \ldots, \psi(\boldsymbol{z}_{|\boldsymbol{Z}_k|})\right)\right]$$

$$\hat{N}_k^{+(n)} = \max_i \rho^{(n)}(i)$$

$$w_{\ell,k}^{+(n)} = \hat{N}_k^{+(n)} \frac{\tilde{w}_{\ell,k}^{(n)}}{\sum_{\ell=1}^{L_k^{+(n)}} \tilde{w}_{\ell,k}^{(n)}}$$

return $\{w_{\ell,k}^{+(n)}, \boldsymbol{m}_{x,\ell,k}^{+(n)}, \boldsymbol{P}_{xx,\ell,k}^{+(n)}\}_{\ell=1}^{L_k^{+(n)}}$ for $n \in \boldsymbol{N}$

4.5.5. Other Solutions. As alluded to in the introductory discussion of this section, this is not the first investigation into using PHD filters for the simultaneous estimation of multiple sets. Presently, a brief comparison between the obtained results and another method within literature, as summarily described in the introduction to Section 4.5, is presented to illustrate the theoretical differences of the methods. As alluded to before, since the methods available in literature are varied and numerous, a representative sample of an approach to these types of multiple-set estimation is presented. In particular, methods similar to the works of [120] and [121] (just to name a pair) are used as a convenient point of comparison. This is to illustrate theoretical differences rather than to indict the referenced works, thus why a representative example is examined rather than pulling from literature directly.

The posterior PHD-product of Section 4.5.3 was obtained via the set derivative of Eq. (4.24), repeated here as

$$\prod_{n=1}^{N} v_k^{+(n)}(\boldsymbol{x}_k^{(n)}) = \frac{\delta^N G_{k|k}}{\delta \boldsymbol{x}_k^{(1)} \cdots \delta \boldsymbol{x}_k^{(N)}}[h_1, \ldots, h_N | \boldsymbol{Z}_{1:k}]\Big|_{h_1 = \cdots = h_N = 1},$$

where $G_{k|k}[h_1, \ldots, h_N|\boldsymbol{Z}_{1:k}]$ is the PGFL of Bayes' rule given by Eq. (4.23). This will be referred to in the following as "Approach A." This set derivative of degree N results in the previously presented corrector equations, and the resulting combinatorics arises from the fact that measurement association ambiguity is introduced and all possible association possibilities must be accounted for to acquire the posterior PHD-product. Then, the n^{th} PHD is obtained by marginalizing out the other $(N-1)$ PHDs.

Instead, one can "directly" computes the marginal of the n^{th} PHD using the set derivative

$$\tilde{v}_k^{+(n)}(\boldsymbol{x}_k^{(n)}) = \frac{\delta G_{k|k}}{\delta \boldsymbol{x}_k^{(n)}}[1, \ldots, 1, h_n, 1, \ldots, 1|\boldsymbol{Z}_{1:k}]\Big|_{h_n=1},$$

where $\tilde{v}_k^{+(n)}(\boldsymbol{x}_k^{(n)})$ is adorned with a tilde to differentiate it from the previous results. In this expression, all of the variables corresponding to the other $N-1$ sets are effectively marginalized out directly by setting all $h_i = 1 \; \forall \; i \neq n$ and then taking a derivative with respect to $\boldsymbol{x}_k^{(n)}$. At first, this may be expected to produce precisely the same thing, but following through with a derivation produces a posterior PHD of the form

$$\tilde{v}_k^{+(n)}(\boldsymbol{x}_k^{(n)}) = \left[q_{D,k}^{(n)}(\boldsymbol{x}_k^{(n)}) + \sum_{\boldsymbol{z}\in\boldsymbol{Z}_k} \frac{p_{D,k}^{(n)}(\boldsymbol{x}_k^{(n)}) g^{(n)}(\boldsymbol{z}|\boldsymbol{x}^{(n)})}{\kappa_k(\boldsymbol{z}) + \sum_{i=1}^{N} v_k^{-(i)} \left[p_{D,k}^{(i)}(\boldsymbol{x}_k^{(i)}) g^{(i)}(\boldsymbol{z}|\boldsymbol{x}^{(i)})\right]}\right] v_k^{-(n)}(\boldsymbol{x}_k^{(n)}).$$

This will be referred to "Approach B" in the following and looks very similar to a standard PHD filter update except for the additional sum over i in the denominator. In fact, this is the same update of the "PHD-like" filters presented in [121], and similar updates can be found in much of the related literature. Note that this can be re-written as

$$\tilde{v}_k^{+(n)}(\boldsymbol{x}_k^{(n)}) = \left[q_{D,k}^{(n)}(\boldsymbol{x}_k^{(n)}) + \sum_{\boldsymbol{z}\in\boldsymbol{Z}_k} \frac{p_{D,k}^{(n)}(\boldsymbol{x}_k^{(n)}) g^{(n)}(\boldsymbol{z}|\boldsymbol{x}^{(n)})}{\tilde{\kappa}_k^{(n)}(\boldsymbol{z}) + v_k^{-(n)} \left[p_{D,k}^{(n)}(\boldsymbol{x}_k^{(n)}) g^{(n)}(\boldsymbol{z}|\boldsymbol{x}^{(n)})\right]}\right] v_k^{-(n)}(\boldsymbol{x}_k^{(n)}),$$

where

$$\tilde{\kappa}_k^{(n)}(\boldsymbol{z}) = \kappa_k(\boldsymbol{z}) + \sum_{\substack{i=1 \\ i\neq n}}^{N} v_k^{-(i)} \left[p_{D,k}^{(i)}(\boldsymbol{x}_k^{(i)}) g^{(i)}(\boldsymbol{z}|\boldsymbol{x}^{(i)})\right].$$

Therefore, this expression can be interpreted as a standard PHD update (thus avoiding the combinatorics of Approach A) but with a pseudo-clutter term $\tilde{\kappa}_k^{(n)}(\boldsymbol{z})$ replacing the clutter term $\kappa_k(\boldsymbol{z})$. This immediately illuminates the difference in methodology: rather than account for all associations as in Approach A, the n^{th} PHD update for Approach B is obtained by treating the returns from all other sets as if they were clutter and otherwise updating according to the standard PHD filter equations.

4.5.6. Numerical Investigations. In the following, numerical investigations are presented to illuminate the characteristics of the newly derived filter. Two scenarios are presented, one that is rather simple and another that is considerably more complex, to demonstrate the key features of this filter. Recall that the new filter that has been derived is referred to as "Approach A" and the comparison methodology described in the previous section (the one that "treats the other targets as clutter") is referred to as "Approach B." After the simple and complex scenarios are discussed, an application to navigation is presented using the reoccurring ballistic trajectory example.

GM implementations are constructed to investigate the similarities and differences in performance. To mitigate growth in filter complexity, merging and pruning, similar to what is described in [69], is applied to the posterior GMs such that components that show 90% agreement according to a χ^2 test are merged using the method of moments and components with weight less than 1×10^{-5} are removed from the mixture. State estimates are obtained in the standard fashion: once the estimated number of targets of the i^{th} set, $\hat{N}^{(i)}$, is computed, the $\hat{N}^{(i)}$ highest weighted components of the i^{th} GM are accepted as state estimates for the i^{th} set.

In each case, 100 Monte Carlo trials are performed with all noises resampled on each trial, and sample statistics are collected across the trials. The following results present the sample means and standard deviations of the localization component of the OSPA metric (seen in Eq. (4.9)) as well as the filter cardinality estimate with $c = 1$ m and $p = 2$. Rather than simply presenting the whole OSPA metric, it is useful in this case to analyze localization and cardinality separately. Rather than using the cardinality component of the

OSPA metric, the posterior cardinality estimates are plotted to assess trends within the cardinality estimation (such as answering the question "Is there a noticeable bias in the resulting cardinality estimate?").

A Simple Scenario. Consider a 20 m $\times$ 20 m, two-dimensional surveillance region containing three types of targets: stationary targets, (nearly) constant velocity targets, and "coordinated turn" targets (somewhat in the manner of [121]). That is, $N = 3$, and three sets are estimated simultaneously such that

$$\begin{aligned} [x \;\; y]^T \in \boldsymbol{X}_k^{(1)} &\rightarrow \text{ stationary target} \\ [x \;\; y \;\; \dot{x} \;\; \dot{y}]^T \in \boldsymbol{X}_k^{(2)} &\rightarrow \text{ constant velocity target} \\ [x \;\; y \;\; \dot{x} \;\; \dot{y}]^T \in \boldsymbol{X}_k^{(3)} &\rightarrow \text{ coordinated turn target} \end{aligned}$$

Note that the difference in state spaces of the sets is naturally accommodated by the presented filter and poses no theoretical or implementational challenges.

In this simple scenario, depicted in Figure 4.35, each set has a single target throughout the 25 second duration of the simulation (there are no spawning, birth, or death events). Note that the trajectory crossing between the constant velocity and turning target occurs about 15 seconds into the simulation, deliberately done to assess if confusion between sets causes apparent strain on the filter. Every second, a position sensor (i.e. measurements are of x and y plus noise) interrogates the surveillance region according to some measurement noise, probability of detection, and clutter profile. The measurement noise is identical for each of the three sets (though the construction of the filter certainly permits these to be different) and is generated according to a zero-mean Gaussian density with standard deviation 0.1 m in both x and y. The probability of detection and clutter profile are altered between two test cases: an ideal case and a challenging case. In all cases, however, clutter is generated uniformly across the surveillance region according to Poisson rate λ, such that the clutter intensity is

$$\kappa_k(\boldsymbol{z}) = \lambda c(\boldsymbol{z}) = \frac{\lambda}{\text{uniform sensing volume}} = \frac{\lambda}{20 \times 20} = \frac{\lambda}{400}\,.$$

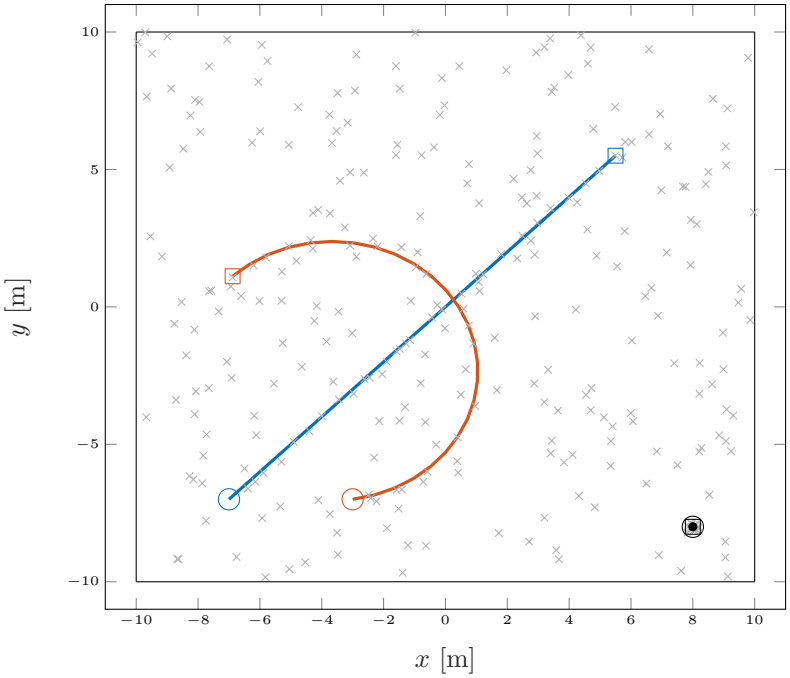


Figure 4.35. Depiction of the simple 20 m × 20 m test scenario. Colors designate target type: black denotes stationary targets, blue denotes constant velocity targets, and orange denotes turning targets. A "∘" denotes the initial state, a "□" denotes the final state, and all "×" are sensor returns.

The three sets evolve according to the following:

1. The stationary target is initialized at $\boldsymbol{x}_0^{(1)} = [8 \quad -8]^T$ and, of course, obeys identity dynamics, i.e. $\boldsymbol{x}_k^{(1)} = \boldsymbol{x}_{k-1}^{(1)}$. The initial state uncertainties are taken as 1 m (1σ), and, since the target is taken to be stationary, there is no process noise (that is, $\boldsymbol{P}_{ww,k-1}^{(1)} = \boldsymbol{0}_{2\times 2}$).

2. The (nearly) constant velocity target is initialized at $\boldsymbol{x}_0^{(2)} = [-7 \quad -7 \quad 0.5 \quad 0.5]^T$ and translates according to

$$\boldsymbol{x}_k^{(2)} = \boldsymbol{F}_{x,k-1}^{(2)} \boldsymbol{x}_{k-1}^{(2)} + \boldsymbol{F}_{w,k-1}^{(2)} \boldsymbol{w}_{k-1}^{(2)} ,$$

where

$$\boldsymbol{F}_{x,k-1}^{(2)} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{F}_{w,k-1}^{(2)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \sqrt{\Delta t} & 0 \\ 0 & \sqrt{\Delta t} \end{bmatrix},$$

$\boldsymbol{w}_{k-1}^{(2)}$ belongs to a process noise sequence generated according to a zero mean Gaussian with covariance $\boldsymbol{P}_{ww,k-1}^{(2)} = 1 \times 10^{-5} \cdot \boldsymbol{I}_{2\times 2}$, and $\Delta t = 1$ second. The initial state uncertainty is taken as 1 m in position and 0.1 m/s in velocity (1σ).

3. The turning target is initialized at $\boldsymbol{x}_0^{(3)} = [-3 \;\; -7 \;\; 0.7 \;\; 0.1]^T$ and translates according to

$$\boldsymbol{x}_k^{(3)} = \boldsymbol{F}_{x,k-1}^{(3)} \boldsymbol{x}_{k-1}^{(3)} + \boldsymbol{F}_{w,k-1}^{(3)} \boldsymbol{w}_{k-1}^{(3)},$$

where

$$\boldsymbol{F}_{x,k-1}^{(3)} = \begin{bmatrix} 1 & 0 & \frac{\sin(\omega\Delta t)}{\omega} & -\frac{1-\cos(\omega\Delta t)}{\omega} \\ 0 & 1 & \frac{1-\cos(\omega\Delta t)}{\omega} & \frac{\sin(\omega\Delta t)}{\omega} \\ 0 & 0 & \cos(\omega\Delta t) & -\sin(\omega\Delta t) \\ 0 & 0 & \sin(\omega\Delta t) & \cos(\omega\Delta t) \end{bmatrix},$$

$\boldsymbol{F}_{w,k-1}^{(3)} = \boldsymbol{F}_{w,k-1}^{(2)}$, the turning rate is set to $\omega = 0.15$, and $\boldsymbol{w}_{k-1}^{(3)}$ belongs to a process noise sequence generated according to a zero mean Gaussian with covariance $\boldsymbol{P}_{ww,k-1}^{(3)} = \boldsymbol{P}_{ww,k-1}^{(2)}$. The initial state uncertainty is taken as 1 m in position and 0.1 m/s in velocity (1σ).

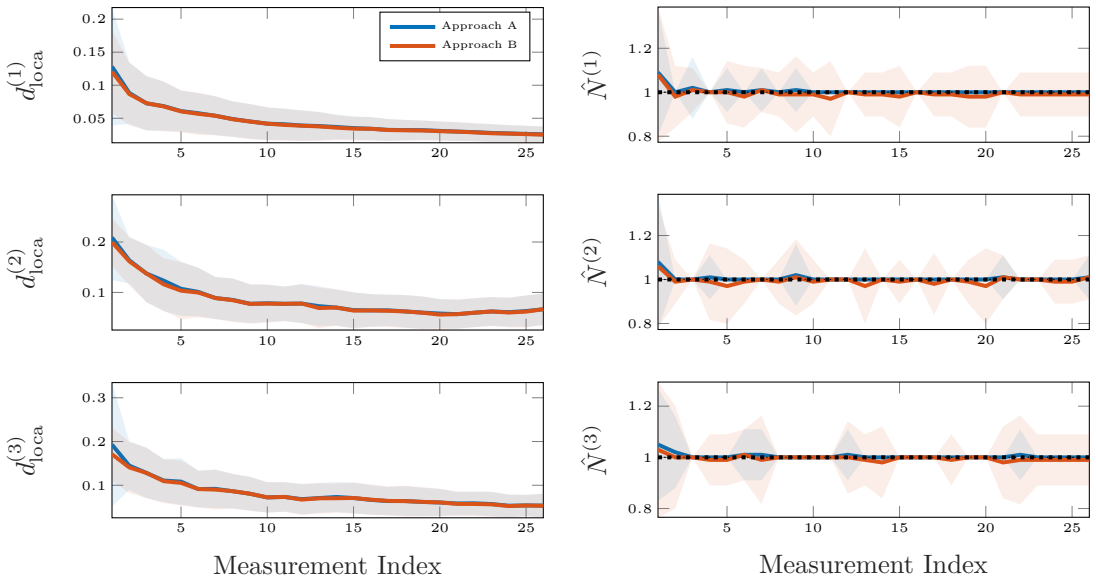
In all of the following trials, initial means are obtained by sampling from a Gaussian centered at the true state according to the initial state uncertainties.

A Simple Scenario: Ideal Case. As a first illustration of performance for the simple scenario, consider the nearly ideal case where the clutter rate is low at $\lambda = 1$ and the probabilities of detection for the three sets are state-independent and high at $p_D^{(1)} = p_D^{(2)} = p_D^{(3)} = 0.99$. This case is "ideal" because the signal-to-noise ratio here is very high, and PHD-type filters are expected to perform very well in such cases.

The Monte Carlo results are presented in Figure 4.36, where Approach A is in blue and Approach B is in orange, and the three rows correspond to sets 1, 2, and 3, respectively. Solid lines represent the sample means over the Monte Carlo trials, and the color-coded shading represents a corresponding 1σ interval around the mean from the sample statistics. Figure 4.36a depicts the localization component of the OSPA metric, and Figure 4.36b depicts the posterior cardinality estimates with true cardinality as dashed lines. Inspection of these results tends toward an immediate conclusion: in this case, Approaches A and B perform nearly identically. In the case of localization performance, the results are nearly indistinguishable. However, close inspection of the shading in Figure 4.36b indicates that Approach A outperforms the cardinality estimation performance of Approach B in that it tends to have a much lower variance in the resulting estimate. This indicates an advantage of Approach A and validates the extra cardinality estimation step required by Approach A over Approach B. The cause of this improvement is perhaps logically attributed to the "flawed" interpretation that enables Approach B in the first place: treating the other targets as if they were clutter. This artificially deflates the signal-to-noise ratio as "seen" by each of the n PHD-like filters used in Approach B and, ultimately, degrades cardinality estimation performance. While this interpretation permits Approach B to maintain linear, rather than combinatorial, complexity, the negative impacts of such an approximation are seen here.

A Simple Scenario: Challenging Case. Now consider a case with considerably lower signal-to-noise ratio where $\lambda = 10$ and each set has much lower, and different, probability of detection, as

$$
\begin{aligned}
p_D^{(1)} &= 0.85 \\
p_D^{(2)} &= 0.70 \\
p_D^{(3)} &= 0.80\,.
\end{aligned}
$$



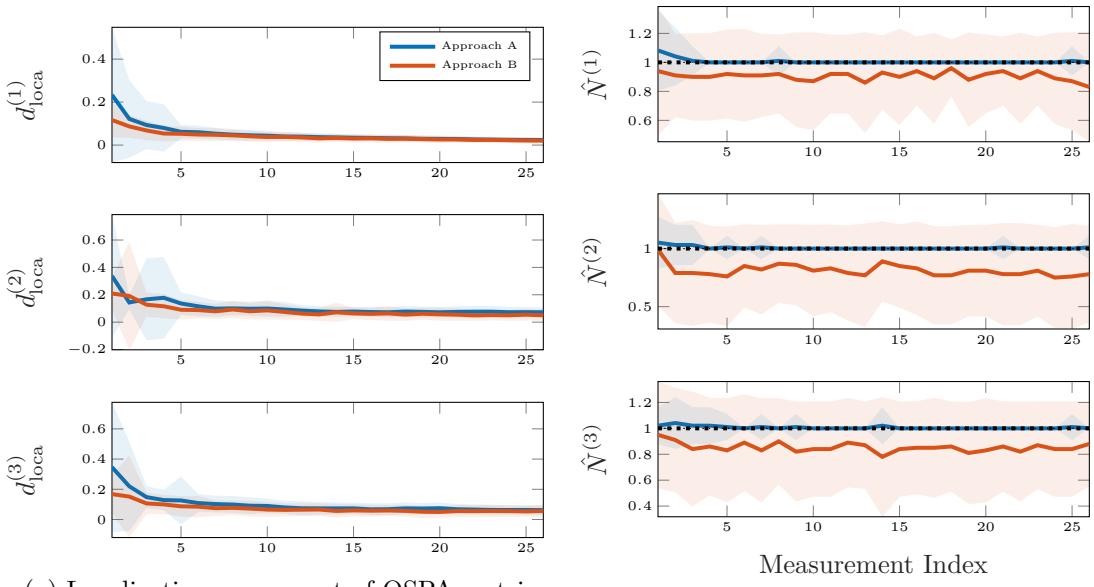
(a) Localization component of OSPA metric

(b) Cardinality estimate and truth

Figure 4.36. Monte Carlo results for the ideal case of the simple scenario. Solid lines are the average statistics and the colored shading indicates 1σ intervals from the sample statistics.

The higher clutter rate is expected to negatively impact estimation performance somewhat, but the truly stressing element on the filter is the much lower probability of detection. It is well known that PHD filters suffer performance degradation as probability of detection drops, and Approaches A and B may be expected to be similarly affected, or even more severely due to the added complexity of estimating multiple sets simultaneously.

In the same manner as the simple case, the Monte Carlo results are presented in Figure 4.37, and some differences begin to appear. As seen in the localization results of Figure 4.37a, Approaches A and B achieve localization performance approximately the same as was seen in the simple case. Approach A degrades slightly toward the beginning of the simulation, but these differences are largely transient, and Approaches A and B begin to perform nearly identically again. A very interesting result is shown in Figure 4.37b, where Approach A exhibits the same strong cardinality estimation performance as it did in the simple case, and Approach B exhibits poorly degraded performance, with a rather strong negative bias in all three sets and a very large sample variance. Ultimately, the artificial



(a) Localization component of OSPA metric

(b) Cardinality estimate and truth

Figure 4.37. Monte Carlo results for the challenging case of the simple scenario. Solid lines are the average statistics and the colored shading indicates 1σ intervals from the sample statistics.

deflation of signal-to-noise ratio by treating the other sets as clutter sources in the PHD-like filters of Approach B is, logically, worsened by the decrease in detection frequency, and the result is severely degraded cardinality estimation performance in Approach B.

A Complex Scenario. Now consider $N = 3$ sets in the same 20 m $\times$ 20 m surveillance region but with a higher population, as depicted in Figure 4.38. As before, the sets, $\boldsymbol{X}_k^{(1)}$, $\boldsymbol{X}_k^{(2)}$, and $\boldsymbol{X}_k^{(3)}$ are estimated as stationary, (nearly) constant velocity, and turning targets, respectively, and the observational scheme is unchanged. However, now there are initially two stationary targets, three constant velocity targets, and two turning targets. Furthermore, spawning and birth is permitted. As marked in Figure 4.38, at index $k = 5$, a constant velocity target is spawned from an existing target, at $k = 10$ a constant velocity target is newly born, and at $k = 18$ a turning target is born. All told, there are initially 7 targets between the three sets, and at the final time there are 10. Otherwise, all system design (such as dynamical models, observation noises, initial statistics, etc.) is kept the same as the simple scenario to preserve continuity in interpreting the results.

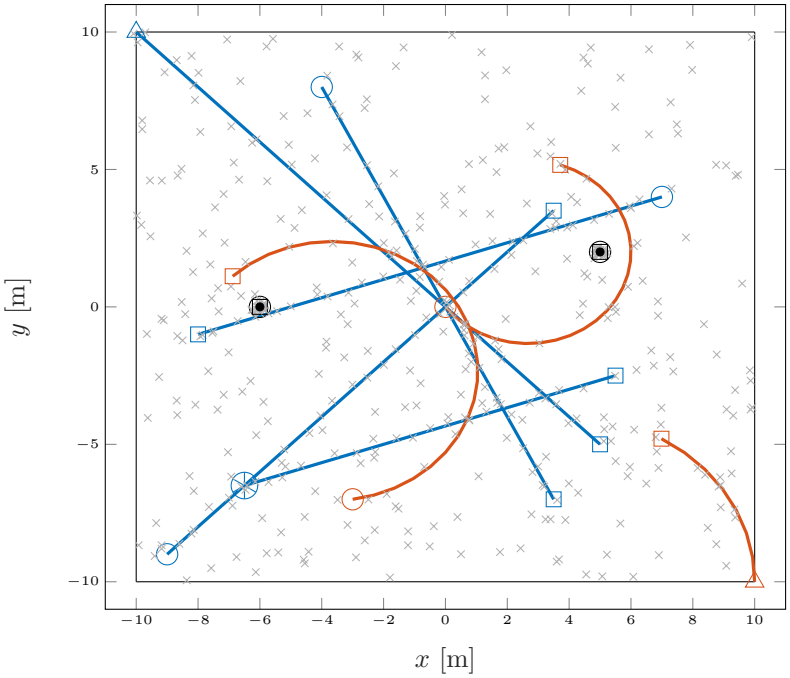
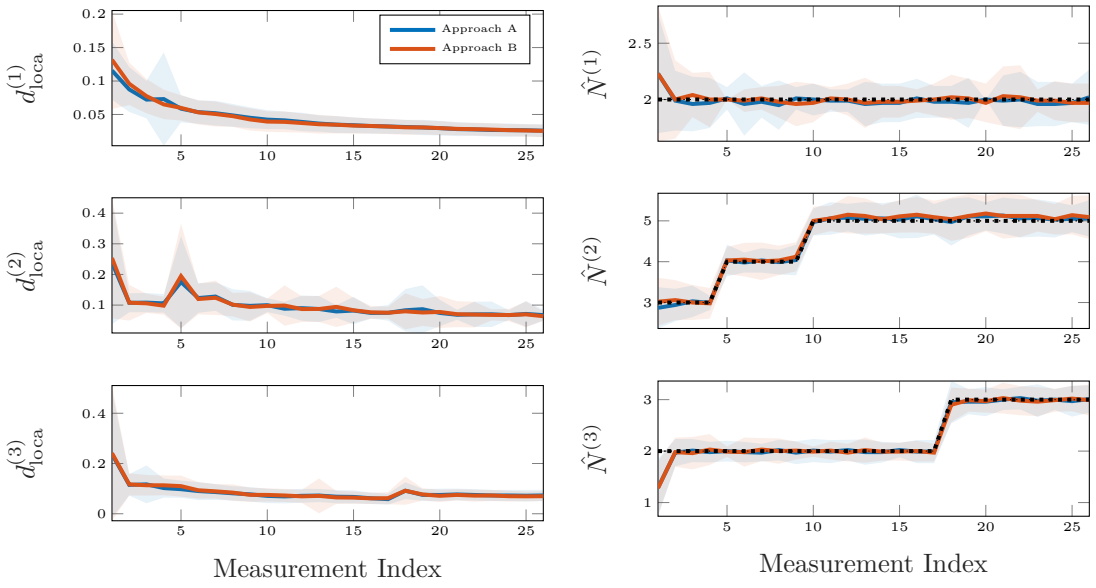


Figure 4.38. Depiction of the complex 20 m × 20 m test scenario. Colors designate target type: black denotes stationary targets, blue denotes constant velocity targets, and orange denotes turning targets. A "○" denotes the initial state, a "□" denotes the final state, a "△" denotes a newly born state, an "∗" denotes a spawning event, and all "×" are sensor returns.

As shown in Figure 4.38, this scenario is notably more complex than the simple scenario and serves to evaluate the impact of target crossings, spawning events, and births on Approaches A and B. The primary congestion event toward the center of the scene is designed such that many of the crossings occur simultaneously to induce any artifacts caused by detection confusion between the targets/sets. Additionally, the two birth events occur at the periphery of the surveillance region to simulate new targets appearing into the scene.

A Complex Scenario: Ideal Case. In the same manner as the simple scenario, first consider a relatively ideal case for the complex scenario: $\lambda = 1$ and $p_D^{(1)} = p_D^{(2)} = p_D^{(3)} = 0.99$. The results for this case are presented in Figure 4.39, where it is immediately apparent that, as in the simple case, Approaches A and B perform nearly identically. The cardinality events (spawning and births) can be clearly seen in Figure 4.39b, and both filters respond quickly to the changes in cardinality. When inspecting the localization errors in

Figure 4.39a, the impact of these cardinality events can be seen as spikes in mean and/or standard deviations at the times of the events ($k = 5, 10,$ and 18). While the two approaches perform nearly identically here, a difference between this and the simple scenario becomes clear. In the ideal case of the simple scenario, the cardinality uncertainty of Approach A was arguably less than that of Approach B, whereas, in this case, performance is nearly identical. It is conceivable that the added complexity of this scenario, particularly due to chances of more confusing detections despite the high probabilities of detection of the ideal case, reduces some of the advantages of Approach A over B.



(a) Localization component of OSPA metric

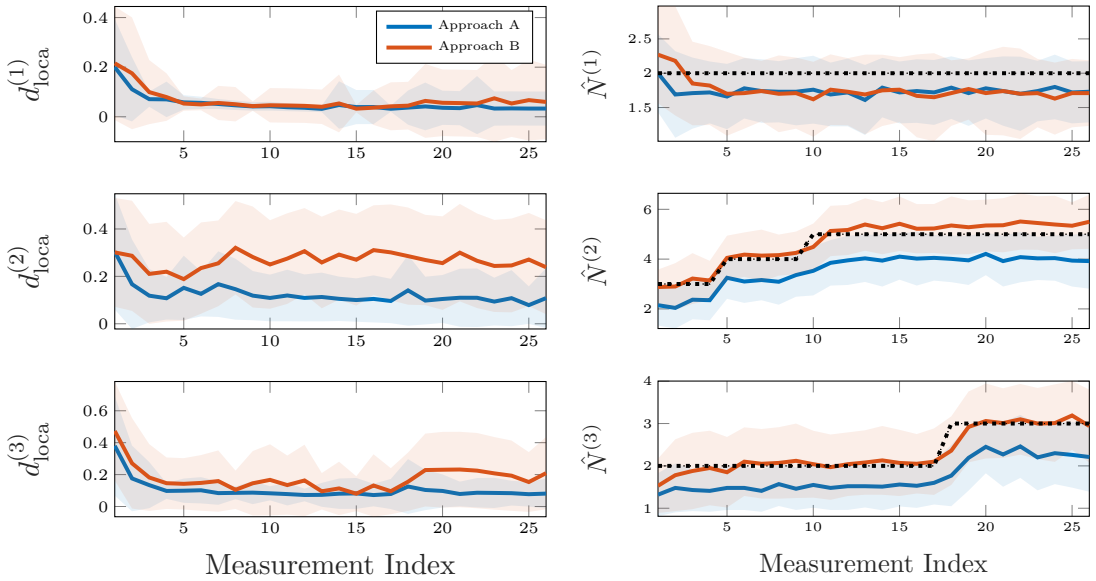
(b) Cardinality estimate and truth

Figure 4.39. Monte Carlo results for the ideal case of the complex scenario. Solid lines are the average statistics and the colored shading indicates 1σ intervals from the sample statistics.

A Complex Scenario: Challenging Case. Now consider a more challenging case where $\lambda = 10$ and each set has probability of detection

$$p_D^{(1)} = 0.85$$
$$p_D^{(2)} = 0.70$$
$$p_D^{(3)} = 0.80\,.$$

The results of this case are shown in Figure 4.40. In this case, Approach A exhibits substantially better localization performance than Approach B, producing localization errors that are both smaller on average (again, a lower solid line) and lower in variance (again, a smaller shaded region). Substantial differences in cardinality estimation performance become visible between Approaches A and B. Approach B is able to rather successfully track the cardinality of the sets as they evolve, but Approach A exhibits a negative bias. The sample variances between Approaches A and B are similar (Approach A's are actually somewhat smaller that of B's), and both appear to track the spawning and birth events, but Approach A's downward bias in cardinality estimate is quite obvious. The reason for this is that the lower probabilities of detection causes additional confusion in the combinatorial sum that accounts for all association and detection possibilities between sets, and, ultimately, the filter is skeptical toward assigning measurements to one set versus the other. In fact, in Swain's dissertation [118], sensitivity studies performed on a related tool that uses a PHD filter for group tracking demonstrate results with a similar downward cardinality bias.



(a) Localization component of OSPA metric

(b) Cardinality estimate and truth

Figure 4.40. Monte Carlo results for the challenging case of the complex scenario. Solid lines are the average statistics and the colored shading indicates 1σ intervals from the sample statistics.

A number of similar experiments have been conducted on Approaches A and B, and the general findings are that Approach A usually exhibits improved localization performance but, as probability of detection drops, it exhibits a downward cardinality bias. As an important counterpoint, Approach B does not suffer from the combinatorial complexity of Approach A and, therefore, might be a preferred tool if runtime is at a premium and a user is willing to sacrifice localization performance. As posed, if λ remains relatively low, Approach A is about 5 times slower on average. This may not be seen as too terrible of a difference for some implementations. As λ climbs, and accordingly the number of measurements to process, this gap in computational performance becomes drastic as the inherent combinatorics take hold. That said, there are a number of potential research avenues for overcoming the complexity of the new filter, such as grouping and assigning measurements via gating to reduce the number of terms in the combinatorial sum dramatically.

4.5.7. Navigation Example. This new filter is applied to the ballistic trajectory navigation example by partitioning the map into two sets as shown in Figure 4.41. Set 1, $\boldsymbol{X}_k^{(1)}$, is said to contain the 5 green features, and set 2, $\boldsymbol{X}_k^{(2)}$, is said to contain the 15 gray features. Other than the partitioning, this is precisely the same map, trajectory, initial conditions, and nominal parameter settings as the numerical studies of Section 4.3.5 on pp. 173. Now, however, the new filter is used to estimate the two portions of the map separately to evaluate if it is feasible for such an application. For example, the green features in Figure 4.41 could be targets of interest, and the gray features may not be of interest but they may be expected to produce sensor returns. In this case, it is particularly interesting to see if one can use only a small subset of the map (in this case, the 5 green features) as "prioritized" features and ignore the remainder of the map. This is somewhat inspired by the concept of the consider filter, but instead "considering" a portion of the map. Can desirable navigation performance be obtained if the new filter is used to estimate $v_k^{+(1)}(\boldsymbol{x}_k)$ but skips updating $v_k^{+(2)}(\boldsymbol{x}_k)$ entirely, i.e. setting $v_k^{+(2)}(\boldsymbol{x}_k) = v_k^{-(2)}(\boldsymbol{x}_k)\ \forall\ k$, to reduce the computational requirements of the filter?

100 Monte Carlo trials are performed to collect performance statistics, and the filter that estimates both sets, denoted "Full," is compared to the filter that only estimates the prioritized features and ignores the remainder of the map, denoted "Prioritized." The

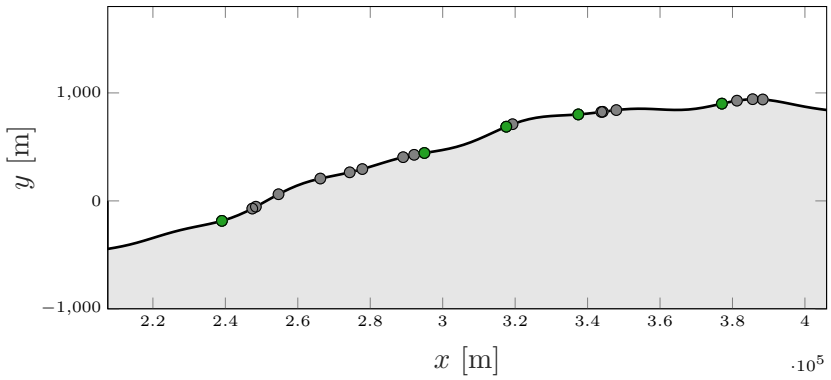


Figure 4.41. Depiction of the partitioned map, where features marked in green are the prioritized features.

navigation performance is presented in Figure 4.42, and the mapping performance is shown in Figure 4.43. Note that the bottom panels of Figure 4.43 contain constant statistics for the second set, and this is because the prioritized feature method never updates the second set's PHD. While the prioritized feature implementation is only marginally outperformed by the full implementation in terms of navigation performance, the biased cardinality estimation of the new filter is exacerbated by neglecting the second set, as seen in the top panel of Figure 4.43b. The bottom panel of the same figure indicates the expected cardinality bias in the full implementation as well. However, it is noted that the prioritized feature method actually exhibits improved map feature localization performance, as shown in Figure 4.43a.

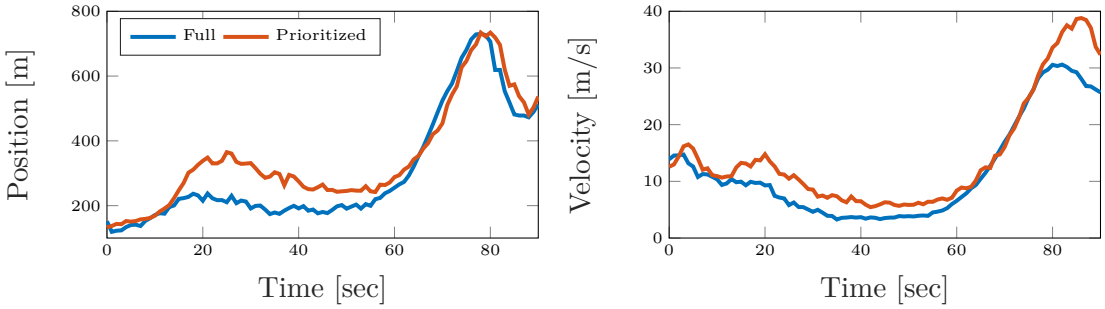
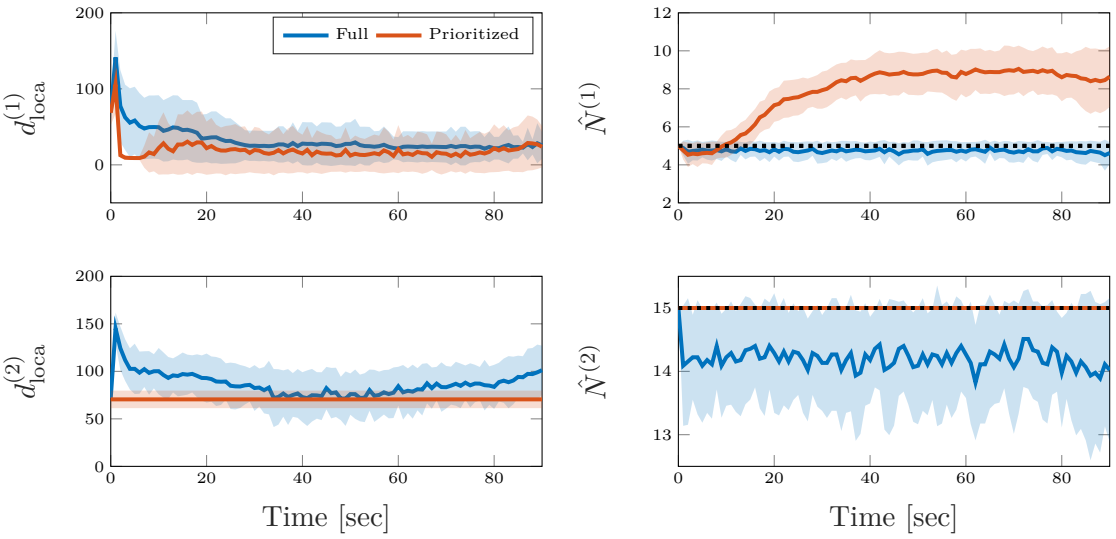


Figure 4.42. Navigation-related 1σ error statistics of the full and prioritized feature methods.



(a) Localization component of OSPA metric. (b) Cardinality estimates.

Figure 4.43. Mapping-related 1σ error statistics of the full and prioritized feature methods.

The key feature of importance here, however, is the timing results presented in Figure 4.44, where it can be seen that the prioritized feature approach requires a mere 2% of the full implementation. This result may at first appear illogical, because it is tempting to assume that ignoring one of two sets should reduce complexity by 50% at best. However, it is important to remember that all PHD-type implementations accumulate extra terms in the GM approximation to the set PHDs to account for missed detection events. By ignoring the second set, these extra terms only accumulate in the first set and, since the first set's cardinality is only a third of the second set's, fewer GM components are required to approximate their positions in the state space. Effectively, this tremendous runtime improvement is obtained by avoiding the accumulated complexity of all 20 map features and, instead, only focusing on the prioritized features. At the same time, while map cardinality estimation substantially suffers, the degradation in navigation performance is modest and map feature localization performance is actually improved.

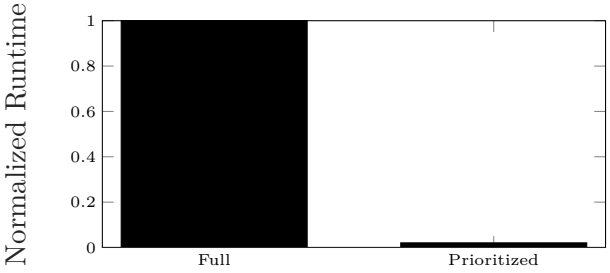


Figure 4.44. Comparison of the normalized runtime required by the full and prioritized feature methods.

5. APPLICATION TO PLANETARY LANDING NAVIGATION

Using cameras for terrain aiding, often referred to as terrain relative navigation (TRN), has been a focus of research for autonomous landing for some time, and [131] presents a thorough survey of related approaches. A prevailing theme among most of this research is the use of sophisticated image processing to avoid explicitly conducting SLAM [109]. The cornerstone of these techniques is matching features within the collected data to known, registered features of some reference dataset, such as a reference image or digital elevation map. In [109], Trawny et al. utilize a technique based on the EKF to combine IMU data and visual data pertaining to *a priori* mapped and registered landmarks, and in [106], Johnson et al. describe many of the salient elements of the visual system's processing. Another related tool is that of APLNav, described by Adams et al. in [132], that stores a map of the reference surface and renders "expected images" for correlation and comparison to those collected by the mobile sensor. The work presented by Alexander et al. in [133] describes the many tools employed by these sophisticated image processing techniques, including but not limited to (i) map rendering and shading, (ii) image normalization, shifting, and scaling, (iii) map and image rectification, (iv) the fast Fourier transform (FFT), and (v) spatial correlation of the image with the reference map. In addition, another theme present in these types of terrain aiding procedures, referred to as "image-correlation" methods in this work, is the use of a past (stochastic) navigation solution to construct the "measurement" provided by the sensor, either as a warm start for the map-to-image correlation methods or to construct an "expected image" of the environment.

The significance of the aforementioned image processing steps and the utilization of a past navigation solution to produce a sensor return, as they pertain to this section, is summarily described as follows: it is likely impossible to model the numerous image processing techniques employed to generate a sensor return for processing in a model-dependent filter, such as the EKF, and, therefore, it is impossible to appropriately quantify the statistics of the collected data and the resulting navigation solutions. Furthermore, any utilization

of the previous navigation solution to produce a processed result necessarily creates correlations between the processed data and the previous navigation solution, correlations that are unaccounted for in all of the image-correlation methods. This section argues that these "measurements," that is, the result of the sophisticated image processing, are not appropriate for processing in a model-based filter like the EKF since they do not produce a mathematical model of the processing that is usable by said filter. Instead, this work chooses to shift the computational burden from advanced image processing to new developments in state-of-the-art multitarget filtering techniques, rather than forcing a datatype into a format that can be crudely interpreted as a traditionally modeled measurement type. The key to this approach is to track the features in collected images as "targets" without requiring knowledge of their identity. The resulting navigation scheme sheds most of the advanced image processing in favor of a well-modeled datatype without any required reliance on an *a priori* reference map and, indeed, returns to treating the landing navigation problem as a SLAM problem.

The SLAM interpretation is compared to the image-correlation methods in Figure 5.1, where special focus is aimed toward the qualitative "cost" bars. The notional costs indicated in Figure 5.1 are not intended to imply that one method is more efficient than the other, but only aim to qualify that image-correlation methods favor complex image processing with traditional filtering whereas the new approach of this work favors advanced filtering techniques paired with much simpler image processing. In a manner not foundationally disparate from the terrain aiding techniques presented in [106, 109, 134], geometric features within an image, such as rocks, craters, and other geographic landmarks, are identified in a collected image with easy to model techniques such as edge detection [135] and centroiding [136]. Instead of taking these features and correlating them to a known reference map, thus introducing impossible to model effects, the features are treated as "targets" within the vehicle's environment. This gives rise to the classic data association problem, and, as described in Section 4, this dissertation posits the use of tools based on FISST as candidate solutions [82, 83]. The FISST-based techniques, utilizing RFSs to model the measurement

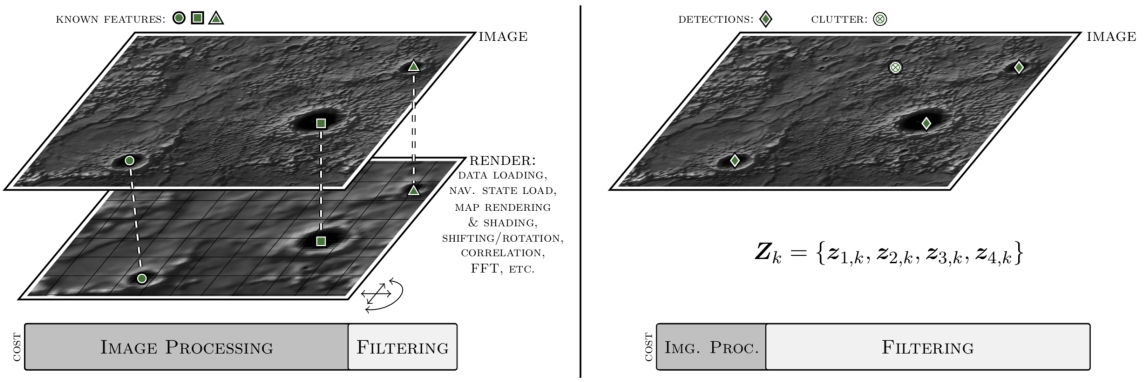


Figure 5.1. Comparison of the image-correlation methods (left) with the methods employed by this work (right). The "cost" bar is meant to signify the relative effort applied to image processing and filtering by each method.

and map features, produce methods that naturally utilize the statistics of these much simpler image processing tools, such as sensor-specific noises and biases, identification failure (probability of detection), and false detection (clutter) rate.

A Motivating Example. An interesting limitation of the sensor suites common to robotic landers is that certain sensors crucial to determining the translational states, such as surface ranging devices like altimeters or velocimeters, do not provide useful information until well after descent is initiated [137]. This means that a navigation filter is relegated to performing uninformative time updates for a lengthy period before obtaining informative measurement updates to the translational states. The result is an inordinate growth in the state uncertainty, resulting in inevitably erroneous guidance and control decisions made by the vehicle during mid-course corrections, pitching maneuvers, and thrusting events. Additionally, these extreme uncertainties promote unstable numerical performance, a prime cause of solution divergence, when the filter is finally able to process the relatively precise ranging data [38, 50, 59].

Consider the descent profile plotted against mission elapsed time (MET) in Figure 5.2. The vehicle initiates descent from orbit at a 50 kilometer altitude, and, after coasting for a period of time while collecting star camera measurements, begins pitching and descent maneuvers at the marker "△". Some time later, the reduced vehicle altitude permits a surface ranging sensor onboard the spacecraft to be activated and collect data,

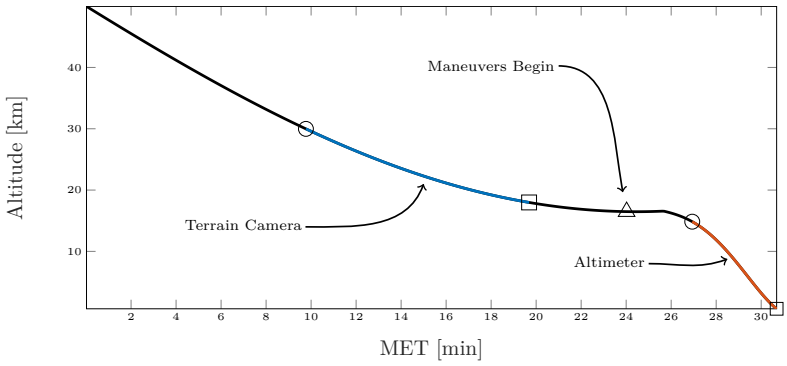


Figure 5.2. Altitude profile of a representative vehicle descent to landing trajectory.

shown in Figure 5.1 starting at a very optimistic 15 kilometer activation altitude. Note that a vehicle state estimate at "△" is bound to be steeped in inaccuracies due to the fact that no position- or velocity-informative sensors have been active. This translates into poorly informed maneuver and ultimately can propagate into large errors in the final landing site or unnecessary expenditures in fuel.

If the vehicle is equipped with a terrain camera, however, the growth in the position and velocity state uncertainties can be mitigated. For example, Figure 5.2 demonstrates a possible terrain-aiding schedule, beginning at 30 kilometers and ending at 18 kilometers, before the thrusting begins. In this situation, translational state information can be collected well before maneuvering is required, leading to improvements in guidance and control decisions and an increased ability to precisely land the vehicle.

5.1. COMMON LANDER SENSING TYPES

The following discussion briefly outlines sensor models for descent profiles like shown in Figure 5.2, but the related discussion is made as concise as possible. This is due to the fact that these details have been presented at length, and, accordingly, references are provided throughout for the interested reader.

The discussion begins by defining the vehicle state as

$$\mathbf{x}_k = \begin{bmatrix} \boldsymbol{r}_k^T & \boldsymbol{v}_k^T & \bar{\boldsymbol{q}}_k^T & \boldsymbol{p}_k^T \end{bmatrix}^T , \tag{5.1}$$

where $\boldsymbol{r}_k$ is a reference position, $\boldsymbol{v}_k$ is a reference velocity, $\bar{\boldsymbol{q}}_k$ is a reference attitude quaternion, and $\boldsymbol{p}_k$ is a vector containing uncertain vehicle-specific parameters, such as sensor biases or dynamical parameters. The reference position, velocity, and attitude are taken to be that of the IMU installed on the spacecraft, a modeling decision that simplifies the dead-reckoning process used for time updates. In this discussion, the subscript "k" is used to denote that a quantity corresponds to time t_k.

5.1.1. Inertial Measurement Units. Modeling the dynamics of a vehicle undergoing powered descent is made drastically complex by difficult-to-model effects, such as drag induced by an uncertain atmosphere, uncertainties in thrust direction and magnitude, and mass property changes in the vehicle over its lifetime. The widely adopted solution to this problem is to replace such models with an IMU and "dead reckon" based on the resulting data; that is, rather than rely on a complex and inherently flawed dynamical model, non-gravitational effects are sensed by an IMU, and these data are used to propagate the vehicle state. This means that, rather than processing the data in a filter that would improve a state estimate, these data are used strictly for time updates and sensing errors effectively become stochastic excitations in the dynamics. While the IMU data are necessarily noise-corrupted, the method is founded on the belief that the IMU error sources are substantially easier to model than the uncertainties in the dynamical system.

IMUs can operate in several ways, such as returning non-gravitational acceleration and angular velocity measurements or returning integrated non-gravitational acceleration and integrated angular velocity measurements. Additionally, internal compensation techniques can be employed to account for coning, sculling, and scrolling effects [138]. In all cases, the output of the IMU is corrupted by a variety of error sources, which include, but are not limited to: startup bias, walking biases due to bias instability, thermo-mechanical noise, scale factor errors, axes misalignments, nonorthogonality of the axes, and quantization [139]. In the current work, the effects of biases, non-orthogonalities/misalignments, and scale factor errors are included in the modeling design; these effects are modeled with a set of parameters that are elements of the vector $\boldsymbol{p}_k$ in the vehicle state. Additionally, it is assumed that the integrated non-gravitational acceleration and integrated angular velocity are output by the IMU.

Using Taylor series to approximate nonlinearities of the IMU dynamics, one can derive a collection of equations that describe the forward evolution of the position and velocity of the IMU and the attitude of the IMU case frame with respect to the inertial frame and their associated error dynamics [137]. In addition to estimating the uncertainties of these position, velocity, and attitude terms, system parameters within $\boldsymbol{p}_k$ can be estimated or, instead, treated as consider parameters [50]. A consequence of using an IMU instead of an explicit dynamical model of non-gravitational effects is that the filter is now subject to processing the immense amount of high-rate data provided by the IMU. Fortunately, the required function evaluations are relatively simple to compute and, with efficient design, permit practical implementation on flight processors [30, 49]. This high-rate data, however, poses a special challenge to the time updates in FISST-based SLAM filters and ultimately motivates the approximation that was explored in Section 4.

5.1.2. Surface Ranging Devices. With time updates handled via IMU-based dead reckoning, other sensors are used to improve the vehicle's state estimate. Surface ranging is one example of a sensor common to most landing vehicles. Here, this is meant to describe the general class of sensors that measure, in some way, the vehicle's relationship to the planetary surface. Perhaps the simplest of these is that of altimeters that, either through barometry or measuring distance along the nadir direction with gimbaled radars or lidars, directly measure the altitude of the vehicle above a modeled reference surface [140]. Another sensor type removes the gimbal and measures the distance along some line(s)-of-sight to the planetary surface, called slant range(s) [140]. A third method utilizes the doppler shift in slant range quantities to resolve information pertaining to the vehicle velocity and altitude [141]. Any or all of these models may contain uncertain parameters, such as sensing biases, that can be included in $\boldsymbol{p}_k$. As alluded to in the previous example, however, a weakness common to all of these sensing types is that they only provide useful information toward the end of a descent profile, often well after maneuvers have been made under large state uncertainties.

5.1.3. Star Cameras. A star camera can be used during quiescent periods of the vehicle descent to resolve a very precise estimate of the orientation of the camera relative to inertial space and, accordingly, the attitude of the vehicle. Star cameras usually operate

by matching the stars present in the sensor FOV to an onboard star catalog and process reference vectors to known stars with an appropriate algorithm, such as QUEST, to produce a quaternion parameterization of the camera frame [37]. As with the surface ranging devices, star camera models can contain parameters such as a biases than can be included in $\boldsymbol{p}_k$. While star cameras do not contribute to improvement in inertial position knowledge, a precise attitude estimate somewhat aids in mitigating the growth of uncertainty in the translational states due to accumulated correlations. When a vehicle is maneuvering, such as during terminal descent, for example, the resulting vibrations typically make the images captured by the star camera useless.

5.1.4. Terrain Cameras. The position vector from the camera to some feature in the planet-centered, planet-fixed frame (denoted by f) is given as

$$\boldsymbol{r}^f_{f/c}(\mathbf{x}_k) = \boldsymbol{\zeta}_k - \boldsymbol{T}^f_i\left(\boldsymbol{r}^i_{\text{imu},k} + \boldsymbol{T}^i_c \boldsymbol{r}^c_{\text{cam/imu},k}\right), \tag{5.2}$$

where $\boldsymbol{\zeta}_k$ denotes the position of a feature in the fixed frame, $\boldsymbol{T}^f_i$ denotes the transformation from the inertial frame to the fixed frame, $\boldsymbol{T}^i_c$ denotes the transformation from the IMU case frame to the inertial frame corresponding to $\bar{\boldsymbol{q}}^{-1}_k$, $\boldsymbol{r}^i_{\text{imu},k}$ denotes the inertial position of the IMU, and $\boldsymbol{r}^c_{\text{cam/imu},k}$ denotes the vector from the IMU to the camera in the IMU case frame. Typically, $\boldsymbol{r}^c_{\text{cam/imu},k}$ will not be time-dependent, but time dependence is included for generality (such as in the case of a pose-controlled camera or displaced cameras). Alternatively, $\boldsymbol{r}^f_{f/c}$ may be expressed in terms of its components with explicit dependence on $\mathbf{x}_k$ as

$$\boldsymbol{r}^f_{f/c}(\mathbf{x}_k) = \begin{bmatrix} x_{f/c}(\mathbf{x}_k) & y_{f/c}(\mathbf{x}_k) & z_{f/c}(\mathbf{x}_k) \end{bmatrix}^T.$$

A measurement, in the form of a pixel coordinate pair, of a single feature is constructed as

$$\boldsymbol{z}_k = f_c \cdot \begin{bmatrix} \frac{x_{f/c}(\mathbf{x}_k)}{z_{f/c}(\mathbf{x}_k)} & \frac{y_{f/c}(\mathbf{x}_k)}{z_{f/c}(\mathbf{x}_k)} \end{bmatrix}^T + \boldsymbol{b}_k + \boldsymbol{v}_k, \tag{5.3}$$

where f_c is the camera focal length, $\boldsymbol{b}_k$ is a bias in the terrain camera measurements (often an element of the parameter vector $\boldsymbol{p}_k$), and $\boldsymbol{v}_k$ is an associated measurement noise with known statistics. The result is a feature-laden image resembling the right panel of Figure 5.1.

The camera-based terrain feature datatype, while rich in information, presents a number of complications. At first, a navigator is tempted to manipulate the incoming measurements to conform with traditional filtering methods, such as the described image-correlation methods. However, even if one is willing to assume that the camera and image processing are perfect, this approach presents a number of challenges/limitations to a standard navigation approach:

(i) Most approaches *require* the use of a known reference map to correlate captured images to [106, 109, 133],

(ii) the approaches that utilize features that are unknown by the filter *a priori* utilize sequential images to reduce *growth* in uncertainty rather than processing the image contents to reduce state uncertainty itself [134],

(iii) without an explicit model for the image processing, there is no way to rigorously quantify and estimate correlation errors,

(iv) construction of the "expected image" is often initialized with the previous navigation state as a warm-start procedure, making the measurement statistically dependent on the previous navigation state [132, 133],

(v) if one seeks to estimate these map errors in an EKF-like filter, broad assumptions must often be made on the errors (such as requiring that all feature errors are caused by a common map-tie error [109]), and

(vi) these approaches limit the vehicle's exploratory authority to remain within the confines of the pre-loaded reference map [109, 131, 132, 133].

A critical drawback of the image-correlation approaches is that they have no way of explicitly modeling false associations, missed detections, or false returns outside of coarse outlier rejection. To combat these issues, instead of forcing this datatype into a Kalman-like

filter, this work augments existing navigation architectures with modern developments in multitarget tracking and SLAM to assimilate terrain camera data into a vehicle's navigation solution.

The author emphatically notes that this discussion is not given in an attempt to invalidate or degrade the exceptional work by the referenced authors in terrain-aided navigation. Indeed, the referenced works, many of them applications with real data produced by hardware test platforms, provide impressive and well-detailed results. The exploration here is motivated by the desire to approach this problem from a different perspective.

5.2. TERRAIN CAMERAS

It is sought to conduct navigation for the landing vehicle using the camera data, where the principal focus is on the vehicle state estimate. In contrast to the image-correlation methods described previously, this approach treats the surface features within the image as "targets" and tracks them directly. The result is a method that utilizes substantially simpler, and easier to model, image processing at the expense of more complex multitarget filtering procedures derived using FISST. The advantage of such a formulation is sevenfold:

(i) The extraction of relevant targets in an image, such as boulders or craters, can be performed reliably and efficiently with edge detection [135] and centroiding algorithms [136], procedures that are substantially easier to model than the advanced image processing required for the image-correlation.

(ii) By modeling the terrain camera as described, all measurement-to-feature correlations are done rigorously within a filter, and, therefore, the resultant filter statistics are well quantified.

(iii) There is no need for a warm-start procedure that creates dependence of incoming measurements to previous state estimates.

(iv) The types of targets that can be tracked are not limited by identity (i.e. boulder vs. crater) as the procedure only requires tracked targets to manifest themselves as significant features within an image (bright, dark, or textured portions of a collected image). By contrast, the image-correlation methods rely strictly on well-tracked and cataloged features, typically craters.

(v) The described methods can accommodate, but do not require, an initial reference map.

(vi) The described methods can accommodate autonomous exploration by utilizing an appropriate data-driven target birth model.

(vii) As described in Section 4.4, these procedures can be used in conjunction with existing, standard navigation architectures in a decentralized fashion, rather then replacing an existing filtering framework.

While this work aims to eliminate the difficult-to-model image processing techniques described in the previous discussion, it is inevitable that some processing must be performed on the collected images. A key observation is that the the missed detection and fault detection rates, handily obtained via Monte Carlo simulation and stress testing of a feature extraction procedure, are explicitly utilized and accounted for in all of the FISST-based filtering mechanisms described later in this section. By contrast, the image-correlation methods have no way of explicitly accounting for these effects. Additionally, since all associations are rigorously modeled by the filter, false associations are not a concern, which is not the case for image-correlation methods.

This work considers extracting the pixel locations of persistent features within a collected image and processing them directly according to the model in Eq. (5.3). The pixel-level bias and measurement noise in Eq. (5.3) come from the offset in the extracted feature from its true location in the image and is another statistic that can be obtained via simulation. So, given the probability of detection, clutter profile, and Eq. (5.3), the measurement is fully and accurately modeled for use in a filter.

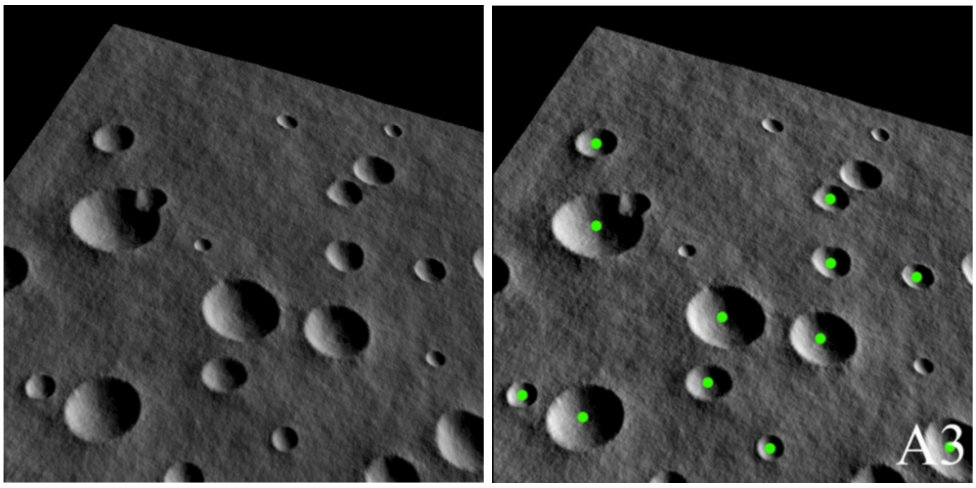


Figure 5.3. Simulated image (left) and feature-detected image (right) from Woicke et al. [142].

Recent work by Woicke et al. in [142] has compared various crater detection algorithms for use in planetary landing and performed extensive analysis to evaluate these methods; an example of a simulated source and processed image can be seen in Figure 5.3. What is important here is that the algorithm that produced the results in Figure 5.3, which is just one candidate algorithm, produces the desired pixel coordinates within the image. Reference [142] presents analyses determining the crater offset (measurement bias and noise), true detection rate (probability of detection), and false detection rate (clutter) for the method, meaning that the required model is complete. It can be clearly seen in Figure 5.3 that some features are missed by the algorithm, but this is not problematic since missed detections can be accounted for directly within the filter.

It may seem odd to claim that image-correlation methods are flawed due to processing, and yet simultaneously propose to use image processing. It is important to remember, however, the critical elements of this approach:

(i) The resulting pixel coordinate measurement of a feature is fully modeled by Eq. (5.3), bias, noise, the detection profile, and the clutter profile;

(ii) a previous navigation solution is never utilized to perform feature extraction or any of the image processing, thereby avoiding the image processing-based state-to-measurement correlations that are ever-present in the image-correlation methods; and

(iii) absolutely no correlation with a known reference map is required for this feature extraction. Instead, a single image is processed according only to its contents and, perhaps, identity-less template craters [142].

The method outlined here for feature extraction is only a representative example. As previously alluded to, the features can be anything that is expected to persist in an image, such as boulders, craters, faculae, maculae, other vehicles, etc.

5.3. AUTONOMOUS EXPLORATION AND BIRTH MODELING

It is inevitable that, when collecting images of a given surface as the vehicle descends, initially untracked map features will generate valid sensor returns. That is to say that, regardless of the fidelity of the initial map with which the terrain aiding mechanism is initialized, it is likely that the camera will return images containing previously untracked features. Furthermore, if a vehicle is to have any exploratory authority in its environment (for example, leaving the previously mapped region in favor of a safer landing site) or if maneuver errors place the vehicle in an unexpected region of the planetary body, these newly observed features should be used to maintain navigation operation. These newly observed features are to be *born* as new map feature estimates and used for subsequent navigation cycles.

First, it must be determined which measurements $\boldsymbol{z} \in \boldsymbol{Z}_k$ are to be treated as corresponding to new map features. A convenient association tool that is built into both the PHD and δ-GLMB filters, and indeed most GM-based estimation techniques, is a term required for the measurement update given as

$$q_{\ell,k}(\boldsymbol{z}) = p_g(\boldsymbol{z}\,;\, \boldsymbol{m}^-_{z,\ell,k}, \boldsymbol{P}^-_{zz,\ell,k})\,,$$

where $\boldsymbol{m}^-_{z,\ell,k}$ and $\boldsymbol{P}^-_{zz,\ell,k}$ are the expected feature-based measurement and associated residual covariance already computed within the filter (of course, one could also use square-root factors). If $q_{\ell,k}(\boldsymbol{z})$ is less than or equal to some specified ϵ for all $\ell \in \{1,\ldots,L_k^+\}$, where L_k^+ is the number of components in the posterior GM estimated by the filter, then $\boldsymbol{z}$ is said to have been generated by a new feature. In the case of a δ-GLMB implementation, the same test is repeated for all ℓ for every hypothesis maintained by the filter (i.e. a birth happens in none of the tracks associated to $\boldsymbol{z}$ according to this test). In practice, this ϵ is quite easily specified as something very "small", and experimentation indicates that, in the case of a birth event, $q_{\ell,k}(\boldsymbol{z}) \approx 0$ for all ℓ, so a user must simply specify that ϵ is a small number, such as 10^{-50}. Given ϵ, new features are born with measurement-dependent probability

$$p_B(\boldsymbol{z}) = \begin{cases} \frac{p_{D,k}(\boldsymbol{x}_k)}{\kappa_k(\boldsymbol{z})+p_{D,k}(\boldsymbol{x}_k)+q_s} & \text{if } q_{\ell,k}(\boldsymbol{z}) \leq \epsilon \ \ \forall \ \ \ell \in \{1,\ldots,L_k^+\} \\ 0 & \text{otherwise} \end{cases},$$

where $p_{D,k}(\boldsymbol{x}_k)$ is the vehicle-map state-dependent probability of detection, $\kappa_k(\boldsymbol{z})$ is the intensity of the assumed-Poisson clutter at t_k, and $q_s > 0$ is a "skepticism" constant that makes the filter more hesitant to trust the appearance of a new object the larger q_s is. This specification of $p_B(\boldsymbol{z})$ accounts for the probability of detection of the sensor while accommodating the possibility that the potential birth return was generated by clutter (through the term $\kappa_k(\boldsymbol{z})$).

What remains is to associate a mean and covariance with a newly flagged birth event, and the method used in this work is graphically depicted in Figure 5.4. An attractive method for instantiating a mean for these newly born map features is to utilize the convenient mathematical relationship for the intersection of a line and a sphere. While the reference surface may not indeed be truly spherical, it may be well approximated as spherical locally. Intersection methods for other reference surfaces, such as ellipsoids or topographical surfaces, are possible but omitted from this discussion.

To that end, note that a unit vector can be recovered from measurement $\boldsymbol{z}$ via

$$\boldsymbol{u}_k^f = \begin{bmatrix} \boldsymbol{z} \\ f_c \end{bmatrix} (\boldsymbol{z}^T\boldsymbol{z} + f_c^2)^{-\frac{1}{2}},$$

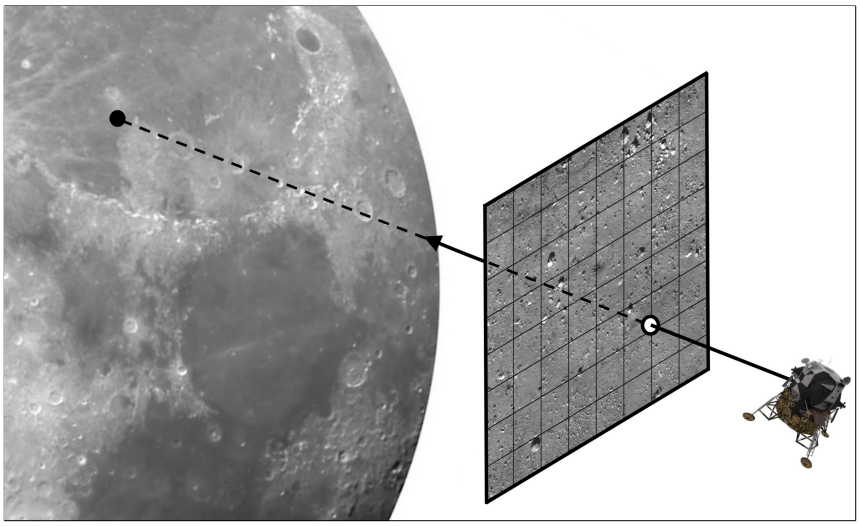


Figure 5.4. Schematic of the birth procedure, a method that projects a point along the line-of-sight of a newly discovered feature to an uncertain reference surface.

since the $\boldsymbol{z}$ are coordinated in the fixed surface frame, where f_c is the camera focal length. Then, computing the term

$$\boldsymbol{r}_k^f = \boldsymbol{T}_i^f \boldsymbol{r}_{\text{imu},k}^i$$

allows the conclusion that, assuming a spherical planetary surface, the intersection point of the line of sight and the sphere defining the planetary body, with spherical radius r_{eq}, is given as

$$\boldsymbol{m}_{\zeta,k} = \boldsymbol{r}_k^f + t\boldsymbol{u}_k^f \,,$$

where

$$t = -\boldsymbol{u}_k^f \cdot \boldsymbol{r}_k^f \pm \sqrt{(\boldsymbol{u}_k^f \cdot \boldsymbol{r}_k^f)^2 - [\boldsymbol{r}_k^f \cdot \boldsymbol{r}_k^f - r_{\text{eq}}^2]} \,.$$

It is intuitive to resolve the "±" by taking the map feature nearer to the vehicle as the solution to the above problem. Then, the spatial density of the target is taken to be Gaussian of the form

$$p(\boldsymbol{\zeta}) = p_g(\boldsymbol{\zeta}\,;\, \boldsymbol{m}_{\zeta,k}, \boldsymbol{P}_{\zeta\zeta,k})\,,$$

where $\boldsymbol{P}_{\zeta\zeta,k}$ is an associated birth covariance that has yet to be specified. Many methods for computing $\boldsymbol{P}_{\zeta\zeta,k}$ can be developed, such as mapping state vector and measurement uncertainties to the reference surface using linearization, but, due to the limitations of space, here it is taken as a user defined value.

This work assumes that the new feature is uncorrelated with the vehicle state at the time of birth. It follows that, according to the principal approximation of Section 4.3, the density describing the vehicle and the newly born map feature is

$$p(\boldsymbol{x}_k) = p_g\left(\begin{bmatrix}\mathbf{x}_k \\ \boldsymbol{\zeta}_k\end{bmatrix}\,;\, \begin{bmatrix}\boldsymbol{m}^+_{x,k} \\ \boldsymbol{m}_{\zeta,k}\end{bmatrix}, \begin{bmatrix}\boldsymbol{P}^+_{xx,k} & \mathbf{0}_{n_x\times n_\zeta} \\ \mathbf{0}_{n_\zeta\times n_x} & \boldsymbol{P}_{\zeta\zeta,k}\end{bmatrix}\right)\,,$$

where $\boldsymbol{m}^+_{x,k}$ and $\boldsymbol{P}^+_{xx,k}$ are the current posterior vehicle state mean and covariance, respectively. Recall that $\mathbf{0}_{m\times n}$ denotes an m-by-n matrix of zeros and n_a denotes the dimension of an arbitrary vector $\boldsymbol{a}$.

Incorporating the newly born feature into the previously estimated map depends on the chosen filter. If an intensity-based filter is employed, such as the PHD filter, the birth intensity is taken as $p_B(\boldsymbol{z})p(\boldsymbol{x}_k)$ and added to the GM representing the vehicle-map state intensity. If a hypothesis-based filter is employed, such as the δ-GLMB filter, the density $p(\boldsymbol{x}_k)$ is taken to be the density of a newly born track with birth probability $p_B(\boldsymbol{z})$. If the utilized filter requires specification of a birth cardinality distribution $\rho(n)$, such as the CPHD and δ-GLMB filters, it is set such that it is equal to $1 - p_B(\boldsymbol{z})$ if $n = 0$, $p_B(\boldsymbol{z})$ if $n = 1$, and zero otherwise. It is noted that, as presented, this birth process accommodates a single birth event at any given time, but it is trivial to "stack" simultaneous birth events based on the association function $q_{\ell,k}(\boldsymbol{z})$.

Countless definitions for a heuristic birth definition, such as the presented one, can certainly be developed. Note that, due to comparing data with the association function to previously tracked targets, this approach technically violates the assumption of independence between surviving and born targets made by some formulations, such as the PHD filter. Previous work within the literature has more formally derived adaptive birth procedures, such as the method in [91] for PHD and CPHD filters, and these methods may be adopted if a mission's computing budget allows it. The method presented here is adopted as a pragmatic engineering solution due to its computational efficiency and intuitive interpretation, and its performance is evaluated in later simulations.

5.4. SIMULATION A: LUNAR LANDING PROOF OF CONCEPT

A Monte Carlo simulation of 100 trials is presented to evaluate the proposed navigation strategy and compare the performance of the ("very simple") PHD and ("very complex") δ-GLMB filtering methods for the application. These methods are used to augment an MEKF, such as described in [37, 112, 137], that is processing IMU, star camera, and altimeter data using fusion with feedback. The described algorithms are applied to the controlled lunar descent trajectory depicted in Figure 5.2 on pp. 248. The vehicle performs pitching maneuvers at about 24 minutes MET and performs continuous descent burns just after 25 minutes MET. The lunar surface beneath the descending vehicle is populated with a collection of point features comprising the true map, and these point features are generated using a simulated digital elevation map (DEM) of the lunar surface.

The simulated lunar lander is equipped with an IMU, an altimeter, a quaternion star camera, and a terrain camera with noise statistics and scheduling described in Table 5.1. Note that the altimeter noise is described as $[100, 10]$ meters. This means that the measurement noise starts at 100 meters (1σ) when the altimeter is activated at an altitude of 15 kilometers and decreases linearly with altitude to 10 meters as the altitude decreases. This is done to coarsely model the noise behavior commonly exhibited by altimeters. Note further that the star camera is assumed to provide measurements only when the vehicle is quiescent, due to the vibrations induced when the vehicle is thrusting.

Table 5.1. Sensor scheduling and noise configuration for the simulated lander.

Sensor	Rate	Noise (1σ)	When active?
Accelerometer	40 Hz	9.81×10^{-5} m/s	Always
Gyroscope	40 Hz	0.1 asc	Always
Altimeter	10 Hz	$[100, 10]$ m	alt. $\leq$ 15 km
Star Camera	1 Hz	30 asc	When *not* thrusting
Terrain Camera	0.1 Hz	1 pixel	When *not* thrusting & 18 km $\leq$ alt. $\leq$ 30 km

Table 5.2. Parameter configuration for the simulated lander. The last column denotes whether or not a parameter is estimated. If not, it is treated as a consider parameter.

Parameter	Mean	1σ (each axis)	Time Constant	Estimate?
$\boldsymbol{b}_v$	$\mathbf{0}_{3\times1}$	9.81×10^{-6} m/s	3600 sec	Y
$\boldsymbol{\gamma}_v$	$\mathbf{0}_{6\times1}$	5 asc	3600 sec	N
$\boldsymbol{s}_v$	$\mathbf{0}_{3\times1}$	175 ppm	3600 sec	N
$\boldsymbol{b}_\theta$	$\mathbf{0}_{3\times1}$	0.05 asc	3600 sec	Y
$\boldsymbol{\gamma}_\theta$	$\mathbf{0}_{6\times1}$	5 asc	3600 sec	N
$\boldsymbol{s}_\theta$	$\mathbf{0}_{3\times1}$	5 ppm	3600 sec	N
b_{alt}	0	2 m	N/A	N
$\boldsymbol{b}_{\text{cam}}$	$\mathbf{0}_{2\times1}$	5 pixels	N/A	N

The parameter configuration, including the simulated parameter statistics, for the simulated lander is shown in Table 5.2. Both the IMU's accelerometer and gyroscope are modeled as having bias, nonorthogonality/misalignment, and scale factor terms, denoted $\boldsymbol{b}$, $\boldsymbol{\gamma}$, and $\boldsymbol{s}$, respectively, where a subscript "v" denotes a term corresponding to the integrated acceleration and subscript "θ" denotes a term corresponding to the integrated angular velocity (see [137]). The terms $b_{\text{alt},k}$ and $\boldsymbol{b}_{\text{cam},k}$ denote biases in the altimeter and terrain camera measurements, respectively. The biases, non-orthogonality/misalignments, and scale factor terms from the IMU are treated as resulting from a first-order Markov process with time constants of 3600 seconds [50]. This is to say that they are modeled as ECRVs with nearly constant dynamics over the approximately 30 minute descent duration. The altimeter and terrain camera biases are taken to be constant. The square-root consider formulation is adopted, where all but $\boldsymbol{b}_{v,k}$ and $\boldsymbol{b}_{\theta,k}$ are treated as consider parameters.

As the vehicle descends, incoming data are processed by an MEKF and the described SLAM mechanisms with the PHD and δ-GLMB filters. Both the MEKF and the SLAM procedures process the incoming IMU data for time updates. Additionally, both the MEKF and SLAM filters employ the aforementioned square-root, consider formulation.[1] The MEKF processes altimeter and star camera data, and the SLAM filters process the pixel-coordinate measurements provided by the terrain camera at 0.1 Hz. It is likely that feature-detected images would be available at a much higher rate [142], but here a "worst case" (i.e. sparse data scenario) is assumed.

The terrain camera is in a fixed position and orientation in the lander's body frame where, put in a simple way that is sufficient for the given discussion, it is pointed "down, ahead, and to the right" of the vehicle's initial attitude. The camera is taken to have a square FOV with a focal length of 5.4 millimeters, an image width of 1920 pixels, and a charge-coupled device width of 5.27 millimeters. The resulting focal length utilized by the terrain camera model is then $f_c = (5.4)(1920)/5.27 = 1967$ pixels. To simulate an imperfect image processing algorithm, features contained in each image generate measurements with probability of detection $p_D = 0.9$ (corresponding to Approach 3 in [142]). To simplify this analysis, clutter returns are omitted.

The true map used in all 100 Monte Carlo trials is illustrated in Figure 5.5, where 17 map features are initialized in the SLAM filters, leaving three completely unknown map features, denoted as "$\square$" markers, that will generate observations and, ideally, generate birth events. The "skepticism" constant (as described in Section 5.3) in determining the birth probability is selected such that $p_B(\boldsymbol{z}) = p_D$, and newly born features have spherical position uncertainties of 30 meters (1σ).

The MEKF is initialized with representative 1000 m position, 0.1 m/s velocity , and 0.573° attitude uncertainties (1σ), and the SLAM filters are initialized with the MEKF solution at the time of the first terrain camera measurement. Recall that initializing with the MEKF solution is non-problematic since conservative fusion is utilized. Each SLAM filter is initialized with $N = 100$ samples of the vehicle trajectory, each sampled according

[1] It is noted that a full covariance, that is, "non-square root", formulation of the consider filters failed on multiple trials for both the PHD and δ-GLMB formulations, indicating the advantages of the square-root *and* consider implementation.

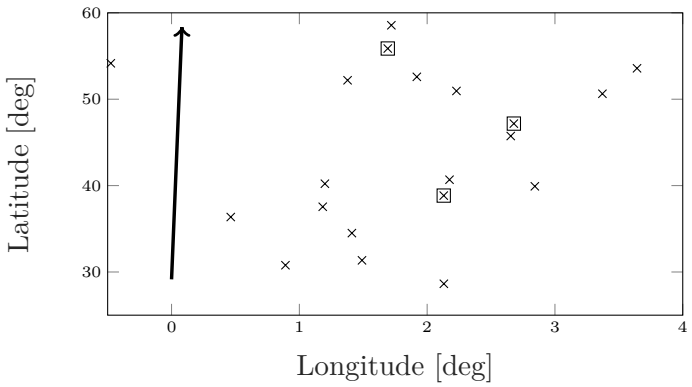


Figure 5.5. Depiction of the true map features, denoted as "×," and the descent trajectory in latitude and longitude coordinates. The three initially uninitialized map features are marked with "□."

to the statistics provided by the MEKF estimate. The map feature estimates are initialized with uncertainties of 10 m (1σ). All initial means, for each of the position, velocity, attitude, parameters, and map feature estimates, are generated by corrupting their true values according to samples from a zero-mean Gaussian distribution with their corresponding statistics.

Fusion is utilized whenever the terrain camera is active, and every time fusion is performed, feedback is used to seed the MEKF and SLAM filter for further processing (see Figure 4.32). Fusion is performed using CI, and the fusion weight is fixed at 0.5 for the duration of the simulation. If the star camera is active, the attitude solution provided by the MEKF is accepted as the fused solution. This is because during most of the descent (until the vehicle begins maneuvering), the MEKF is processing star camera data and therefore has a very accurate attitude estimate, whereas the SLAM filters, as formulated, have no access to this highly accurate information (note that, as mentioned in Section 4.3.3 on pp. 169, one could also process this data within the SLAM filters).

Both the PHD and δ-GLMB filters have a number of maintenance methods that keep model complexity feasible through time. In both, GM components are merged according to a 90% agreement test, and any component with a weight below 1×10^{-5} is removed from the mixture. In the δ-GLMB filter, any hypothesis with weight below 1×10^{-3} is removed, and the weights of the remaining hypotheses are normalized such that they sum

to one. As mentioned before, Reference [73] describes schemes to truncate the number of birth, surviving, and updated hypotheses produced by the δ-GLMB recursion, and this work uses these methods to only explicitly form the top 5, 10, and 10 weighted hypotheses, respectively. Intuitively, birth hypotheses are only generated in the event that $p_B(\boldsymbol{z}) > 0$.

Simulation A: Results. To assess the performance improvements of the proposed approach to terrain aiding, the fused solutions from both the PHD and δ-GLMB filters are compared to the standard MEKF that does not process terrain data at all. These results are not compared directly to the image-correlation methods described in the introduction to this section because, as depicted in Figure 5.1 on pp. 247, the images are used in completely different ways. In essence, the dataypes are different for each of the interpretations, making the comparison "apples-to-oranges" with respect to terrain aiding techniques. Therefore, no such comparison is pursued.

The sample error covariance over the 100 Monte Carlo trials are computed for both the unfused MEKF case and the fused PHD and δ-GLMB solutions. The sample root-sum-square (the square-root of the trace of covariance) for position, velocity, and attitude are shown in Figure 5.6 and are interpreted here as 1σ intervals of their corresponding state variable. Additionally, the position and velocity 1σ intervals are plotted in the UVW frame (where u is the radial direction with respect to the lunar center, v is the vehicle's in-track direct of motion, and w is the vehicle's cross-track direction) in Figures 5.7 and 5.8. In all of these plots, the points at which the terrain camera starts and stops producing data are denoted by "$\circ$" and "$\square$", respectively, and the time at which the powered descent begins is denoted by a vertical line. Note that these markers are consistent with the trajectory shown in Figure 5.2.

An enhanced view of the altitude channel (that is, u) is shown in Figure 5.9. These results indicate that at the end of the trajectory, the vehicle's altimeter dominates the altitude channel since all of the solutions are virtually the same. It is possible that performing additional terrain aiding toward the end of the trajectory, rather than or in addition to using terrain cameras at higher altitudes, could improve these results, but it is these lower altitudes that are particularly well-suited for surface ranging devices such as altimeters.

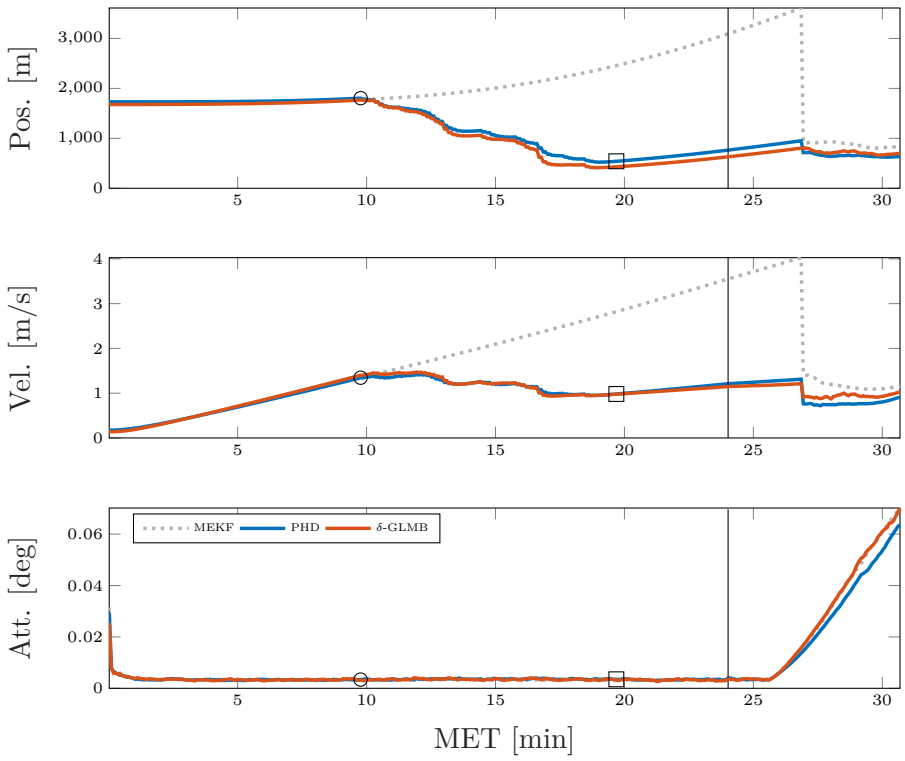


Figure 5.6. Norm position, velocity, and attitude Monte Carlo root-sum-square statistics, here interpreted as 1σ intervals.

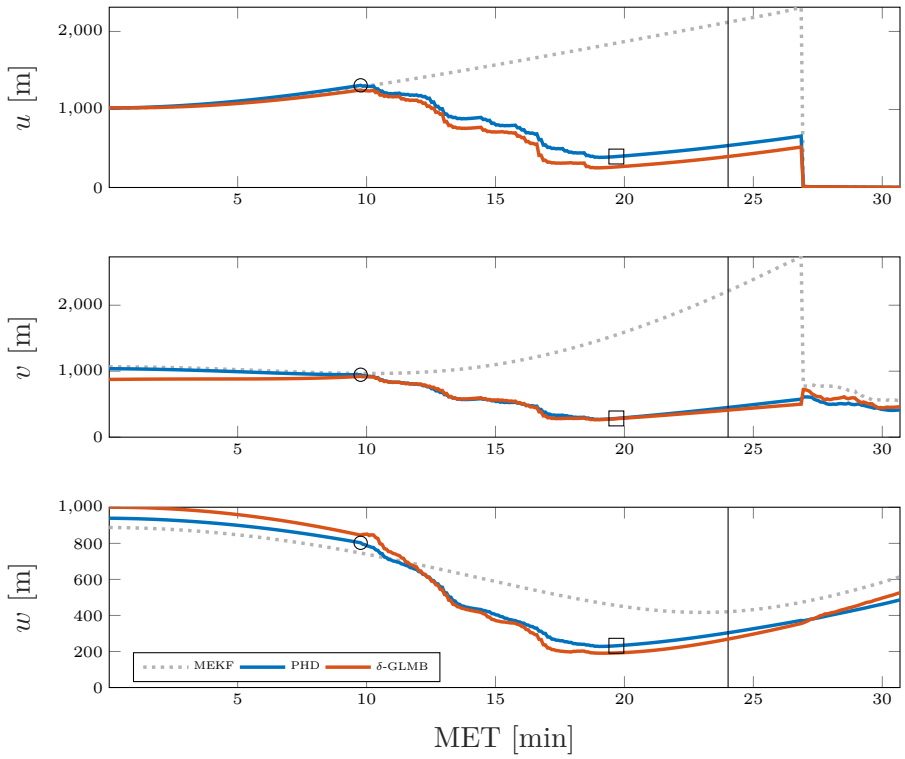


Figure 5.7. Position Monte Carlo 1σ intervals in the UVW frame.

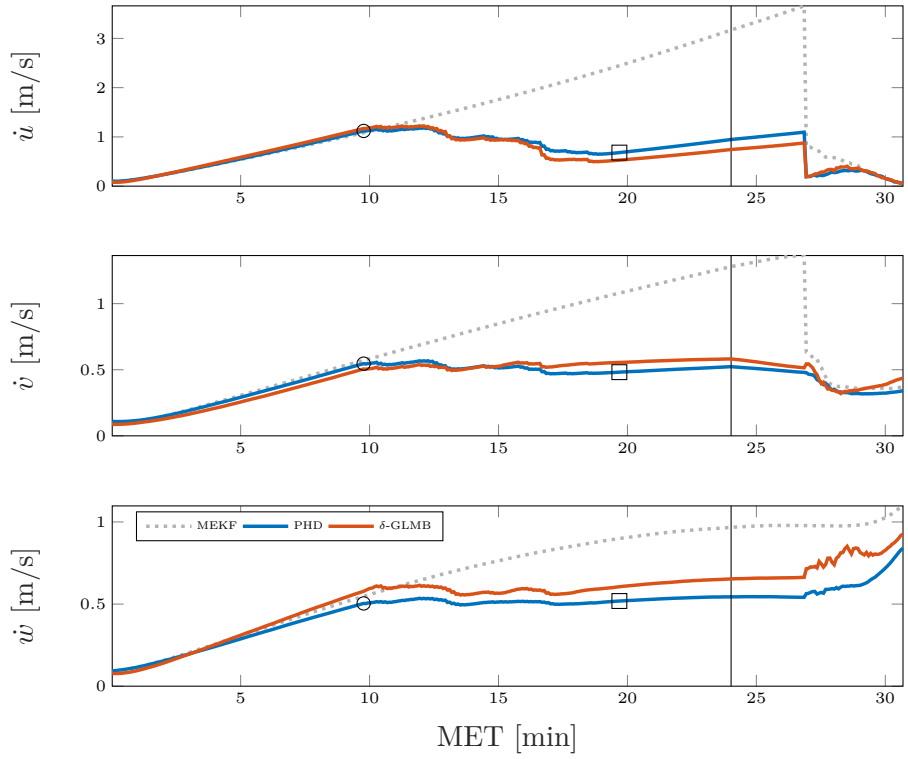


Figure 5.8. Velocity Monte Carlo 1σ intervals in the UVW frame.

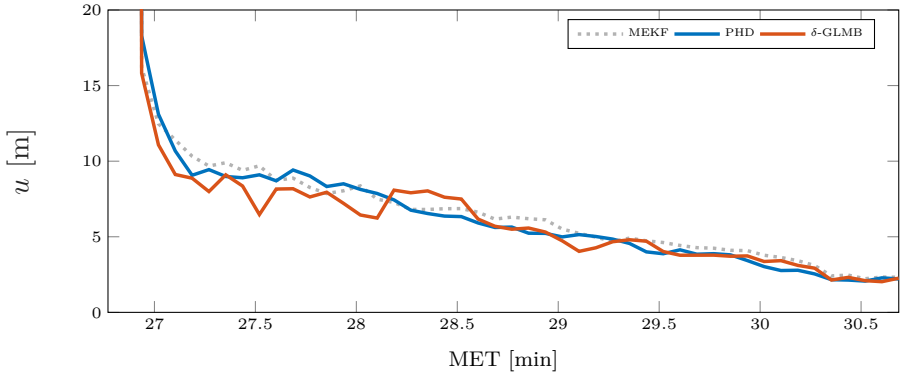


Figure 5.9. Enhanced view of the final minutes of the position Monte Carlo 1σ intervals for the u (altitude) channel.

Note that fusion does not impact the attitude estimate in any meaningful way. This is because, despite the correlations that exist between the map and vehicle attitude, the precise attitude information provided by the star camera dominates the attitude estimate. When the vehicle begins its descent thrusting, the star camera is forced to turn off, and thus attitude uncertainties increase. However, the rate of increase is manageable and the final attitude estimate remains well below 1° (1σ).

The improvement in state uncertainty is intuitive, as the inclusion of data should always yield improved results, but this serves as a motivating example of just how impactful terrain data can be. This is an open loop simulation of descent along a reference trajectory rather than a fully closed-loop simulation (that is, with guidance, navigation, and control in-the-loop), but note that the vertical line in each plot as the point at which the simulated vehicle would need to start making autonomous guidance and control decisions. These results indicate that terrain aiding allows drastically reduced state uncertainties at this point and on, intuitively translating into better guidance and control decisions due to improved navigation estimates. When the vehicle's altimeter is finally able to begin collecting data (plainly seen as the sharp decrease in the MEKF position and velocity uncertainties in Figure 5.6), the vehicle would have already made maneuvering decisions using state estimates that are four times as uncertain as the fused solutions.

These results also corroborate one of the most striking conclusions of Section 4.3: despite the vast differences in theoretical complexity, using the current approximation, the PHD and δ-GLMB filters produce very similar state estimates. There are slight differences, with Figure 5.6 indicating that δ-GLMB has improved position estimates and PHD has improved velocity estimates at the end of the simulation, but these differences are most likely induced by sampling errors. Even with the varying number of map features, the vehicle state estimates are continually improved, indicating that these methods are suitable for missions that require autonomous exploration outside of the confines of a reference map.

Where the differences lie in these methods is in mapping performance. The Monte Carlo statistics of the OSPA metric, here used to measure the error in map estimates, are shown in Figure 5.10 (using OSPA parameters $p = 1$ and $c = 50$). The OSPA trends make

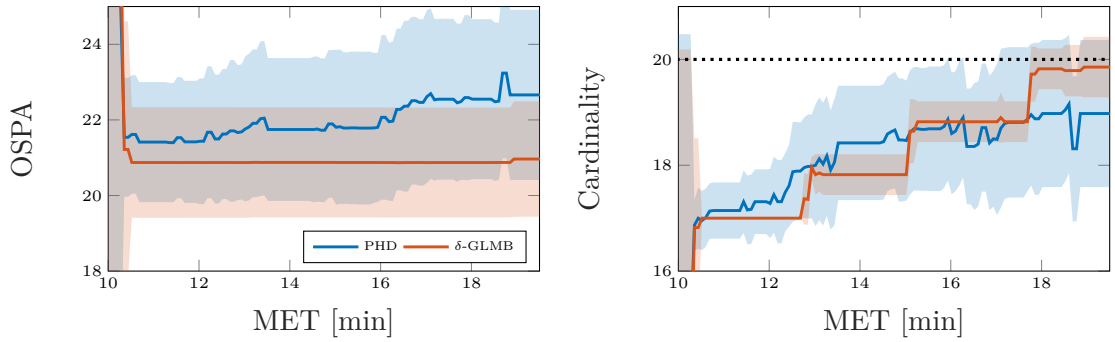


Figure 5.10. Monte Carlo statistics for the OSPA metric and cardinality estimates with sample mean as a line and the shaded region indicating its 1σ interval.

it clear that the δ-GLMB filter produces map estimates with lower OSPA values, and thus lower errors. It turns out that the PHD filter's larger mean OSPA (line) and OSPA variance (shaded region) is due principally to its cardinality estimate.

The cardinality results in Figure 5.10 indicate that both filters are able to track the birth of the *a priori* unknown features, but, as anticipated, the δ-GLMB produces more accurate cardinality estimates with lower variance. The PHD filter underestimates the final map cardinality on average, but the larger variance envelops the true cardinality of 20 features. The histogram in Figure 5.11 depicts the cardinality estimates by both filters when the terrain camera turns off for all of the trials. This figure corroborates the fact that the δ-GLMB filter produces more accurate cardinality estimates with a lower variance, but it also indicates that the PHD filter's trials clearly have a mode at the true value of 20 features (whereas the sample mean indicates an estimate of 19) meaning it is working precisely as it should.

A consequence of the PHD filter's formulation is that its cardinality estimate's variance scales with cardinality, and thus, as can be seen in Figure 5.10, this variance increases as new features are discovered. This means that, with respect to map quality, the PHD filter may be an unattractive option if many birth events or large numbers of map features are expected. However, in informal timing studies of the trials with unoptimized code, the δ-GLMB filter runs took 192 times as long on average as the PHD filter runs. Essentially, this corroborates the findings of Section 4.3, that a navigator may choose any

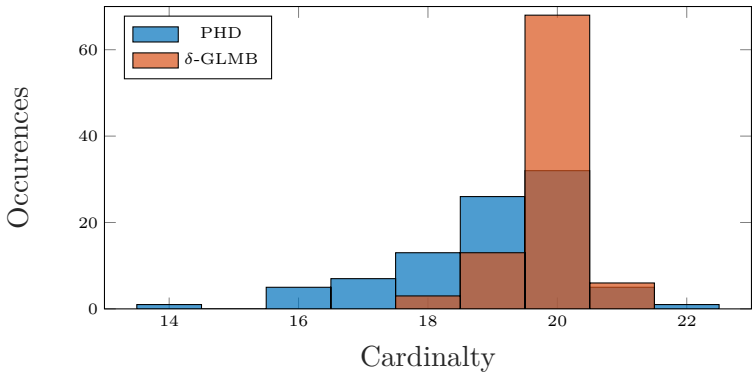


Figure 5.11. Histogram depicting the filters' cardinality estimates when the terrain camera turns off for all 100 Monte Carlo trials.

FISST-based filter that meets their mapping requirements, since this bookend-style comparison of "simple" and "complex" filters indicates that vehicle state estimation is relatively unaffected. While the δ-GLMB filter produces improved mapping results, it is most likely that the CPHD filter [70, 88] or the newly-developed second-order PHD filter [89] represent attractive compromises between map estimation and computational cost. Alternatively, an approximation of the δ-GLMB filter, known as the labeled multi-Bernoulli filter may be an attractive option, particularly if the flight processor has parallel computing capabilities [99].

5.5. SIMULATION B: REALISTIC LUNAR MAP

To further assess the capabilities of the proposed approach, a map is designed using real lunar data, taken from The Unified Lunar Control Network (ULCN) published by the U.S. Geological Survey.[2] The ULCN is a collection of some 272,931 well-localized points on the lunar surface that have been logged using data provided by the Apollo, Mariner, Galileo, and Clementine missions. This dataset provides access to real examples of the types of features that would be tracked using terrain-aided navigation techniques, and permits further insight into the proposed approach. While the previous simulation was conducted using a fully simulated map, the current simulation considers the use of the ULCN as a source of real data.

[2]The ULCN data was obtained from https://pubs.usgs.gov/of/2006/1367/.

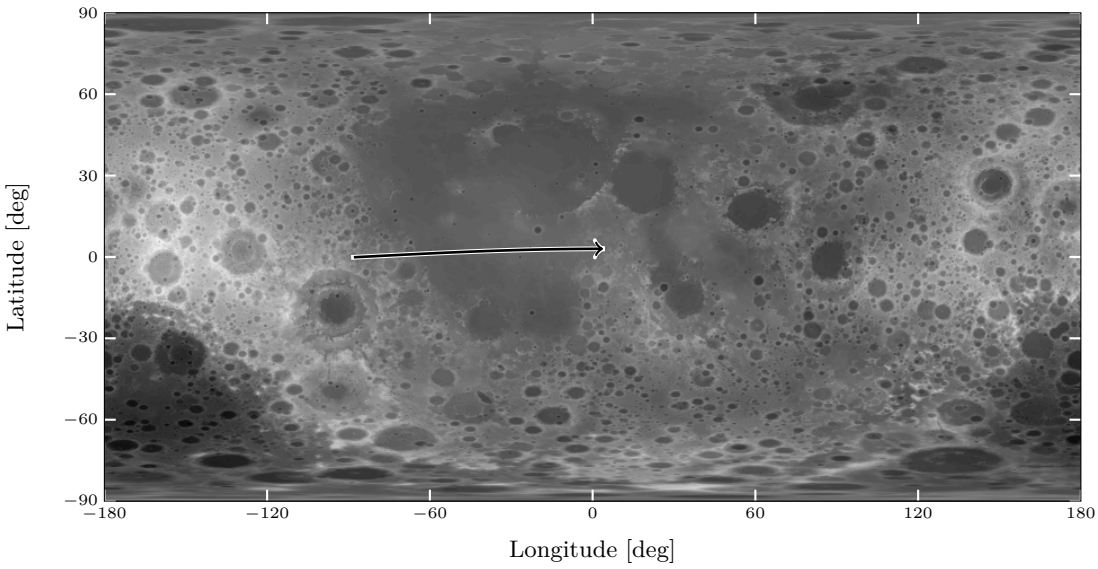


Figure 5.12. The portion of the vehicle trajectory that the camera is on plotted above the lunar surface.

Consider the same trajectory profile of the previous example (as depicted in Figure 5.2 on pp. 248) using all of the same parameters, state vector, consider parameter vector, noise statistics, probability of detection, etc. Now, however, the terrain camera is active for a longer duration, from 30 km to 10 km altitude, and the vehicle is said to traverse the lunar surface as depicted in Figure 5.12, where the terrain information shown is taken from the DEM provided by the Lunar Orbiter Laser Altimeter (LOLA) mission.[3] The portion of the trajectory shown in Figure 5.12 is only the part of the trajectory when the terrain camera is on, and the corresponding map features that the terrain camera sees during this portion of its descent is shown in Figure 5.13. In this figure, the top-most panel depicts the whole of the surface and corresponding features observed by the camera,[4] and the remaining panels subdivide the whole map into three different subsections of longitude to show detail. All told, there are 445 lunar map features taken from the ULCN, considerably more than the 20 considered in the previous simulation.

[3]The author would like thank Kari Ward for assisting in interfacing with the LOLA DEM datafiles and to Dr. Kyle DeMars for providing the capability to utilize the ULCN data.

[4]This is why only a portion of the trajectory is shown in Figure 5.12.

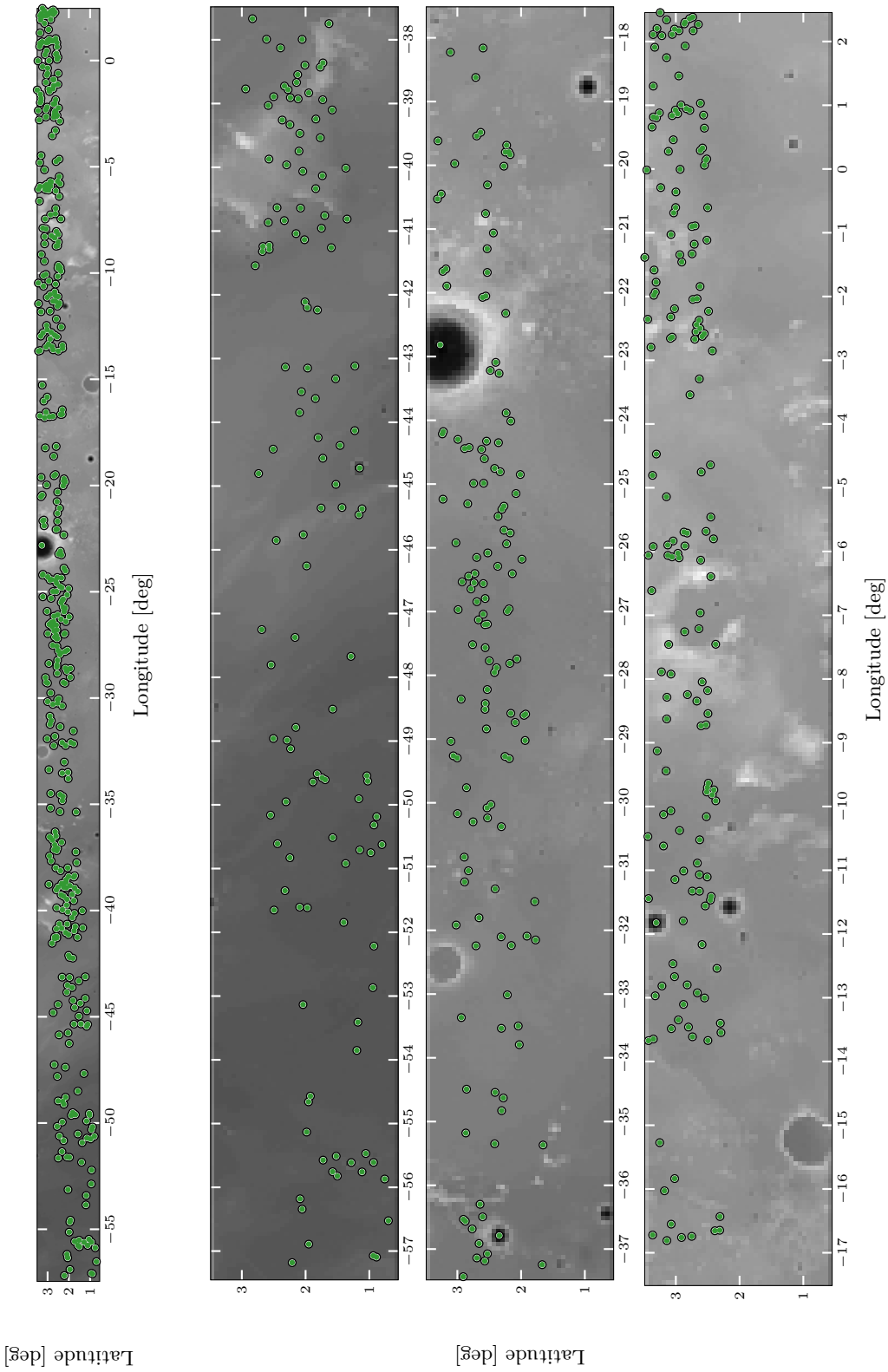


Figure 5.13. Depiction of the 445 lunar map features observed by the terrain camera.

The reasons for extending the terrain camera window from altitudes of [30,18] km to [30,10] km are threefold. First, this selection permits a larger overall map size, since the camera is allowed to observe for a larger portion of the trajectory, and this should stress-test the mapping performance of the filter. Second, it permits observations to be taken closer to the ground to determine if geometry-related issues present themselves (such as those seen in Section 4.3.5). Third, it permits the terrain camera and the altimeter to be active simultaneously, such that any sensitivities to simultaneous processing of the fused filters can be observed.

The position, velocity, and attitude 1σ intervals resulting from 100 Monte Carlo trials are presented in Figure 5.14, and the mapping performance is presented in Figure 5.15. Only the PHD filter is shown due to the agreeable performance it has exhibited throughout this dissertation. As before, the PHD filter is implemented with the square-root consider formulation, and decentralized fusion is used to augment a square-root consider MEKF processing altimeter and star camera measurements.

Figure 5.14 indicates that, as seen previously, the terrain aiding strategy is capable of utilizing the terrain camera data to improve state estimates before maneuvers occur. As before, due to informative star camera measurements, the difference between the MEKF-only case and PHD filter with fusion are negligible. In this case, terrain aiding permits nearly a halving of position and velocity uncertainties before maneuvers begin, but it is observed that the drastically increased map size somewhat inhibits the PHD filter's ability to improve state knowledge. This indicates that a navigator should carefully consider the design of the employed map, and that a lower map density should result in ideal navigation performance, with the additional advantage of reduction computational burden. While this does indicate that the selection of the *a priori* map has a significant effect on the navigation performance of the approach, as is logical for any such approach, it is important to note that, in contrast to the image-correlation methods, this can theoretically operate without prescribing *any* initial map at all. It is possible that starting a map "from scratch" onboard could be a feasible solution for meaningfully obtaining navigation solutions with a low computational burden, and this is an avenue for further research.

The mapping performance presented in Figure 5.15 indicates that increasing the map size by so much (more than 20 times the number of features) degrades the PHD filter's map estimates, unsurprisingly due to its cardinality estimation. On first inspection of the OSPA trends, it appears that the filter is entirely unable to manage the *a priori* map, but it is important to remember that the OSPA metric penalizes errors in both localization and cardinality estimates. Due to the PHD filter's Poisson cardinality modeling, it is obvious that as target number increases, so does its cardinality variance, ultimately degrading the cardinality estimation performance of the filter. This is plainly observed in the right panel. However, the left panel additionally shows the localization errors from the map, computed as the term d_{loc} in Eq. (4.9) on pp. 150, that indicate the filter is actually able to maintain map fidelity throughout the mapping period, and the increase in OSPA is almost entirely due to the degraded cardinality estimates. The cardinality estimated by the filter appears to decrease through time, but considering that, for a mean cardinality of 445, the corresponding cardinality standard deviation is $\sqrt{445} \approx 21$, the minimally achieved mean estimate of 431 is well within 1σ. This is important because, while cardinality performance suffers, the statistics remain well-quantified.

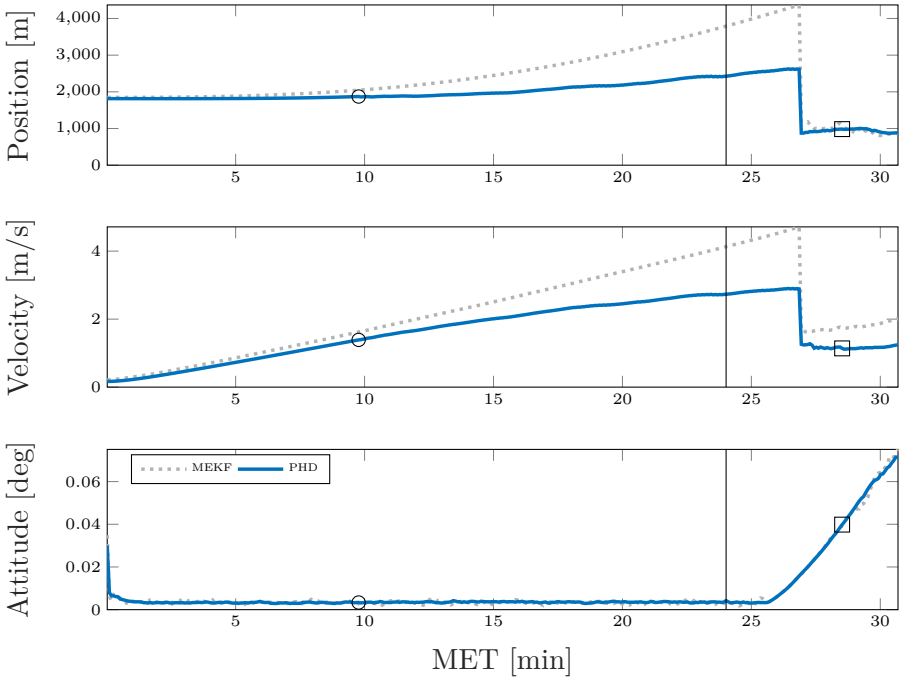


Figure 5.14. Norm position, velocity, and attitude Monte Carlo root-sum-square statistics, here interpreted as 1σ intervals.

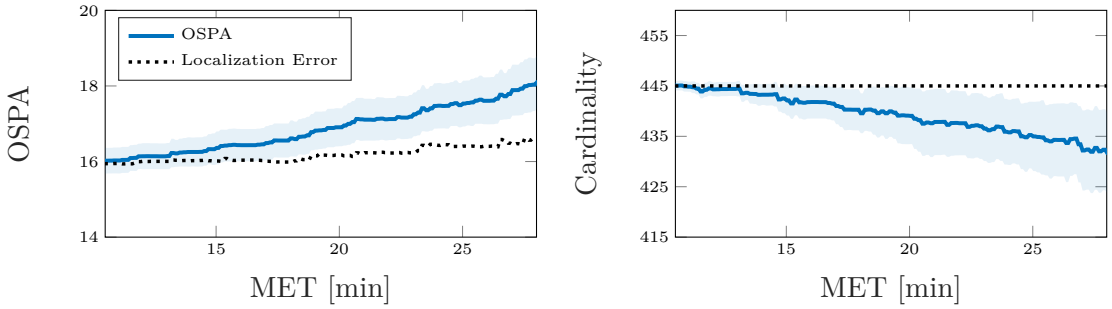


Figure 5.15. Monte Carlo statistics for the OSPA metric and cardinality estimates with sample mean as a line and the shaded region indicating its 1σ interval.

6. CONCLUSIONS

This dissertation has focused on techniques enabling numerically stable filtering implementations for target tracking and navigation, and a strategy for conducting terrain-aided navigation in challenging environments was presented. Section 2 delivered background on standard approaches to minimum mean square error (MMSE) estimation, discussed a number of practical navigation techniques that promote filter stability, and presented equations defining square-root formulations of the standard filters.

In Section 3, the equations for the square-root consider filter were presented, following a discussion on the numerical technique that enables them, the hyperbolic Householder reflection. After the square-root consider filter was derived and detailed for both linearization- and quadrature-based applications to nonlinear systems, a numerical study verified the produced results, and its stability improvements over standard methods were demonstrated. Following these developments, corollary implications for standard (i.e. non-consider) square-root filters were discussed. In the interest of enabling real-time application, a specific set of modeling decisions were presented that enable drastic performance improvements, and timing studies indicate that the efficient implementation can offer a many order of magnitude reduction in required computing time, particularly for high-dimensional problems.

Section 3 continued by deriving the consider filter using Bayes' rule, rather than the historically-cherished MMSE interpretation. The implications of this result were explored, and the result was used to derive the Gaussian mixture (GM) consider filter that, rather than working with mean and square-root factor, parameterizes state uncertainty with a weighted sum of Gaussian components. The new filter was compared to the previously-studied square-root consider filter, and the relevant differences were discussed. In particular, the GM consider filter was found to share the steady-state performance of the filter that only uses mean and square-root factor, but the GM consider filter's transients exhibit more desirable behavior, i.e. lower estimation error uncertainties.

Section 4 directly leveraged the previous results to derive the GM consider probability hypothesis density (PHD) filter, effectively generalizing consider filtering to the multitarget domain. Sufficient details were provided such that an interested reader could use them to produce consider-analogs to many of the common multitarget filters in literature, such as the cardinalized PHD (CPHD) filter, the multitarget, multi-Bernoulli filter, and the δ-generalized labeled multi-Bernoulli (δ-GLMB) filter. The newly discovered GM consider PHD filter was numerically implemented, and the results indicated an interesting feature of parameter estimation using PHD filters. Since each of the targets estimated by the PHD filter share the same state space, if parameter common to all the targets is estimated, rather than considered, it is possible that, due to problem geometry and errors in approximation of the nonlinear system of interest, a multi-modal estimate for a given parameter is possible even if it starts as, and is expected to remain, Gaussian. This caused substantial degradation in performance of the standard PHD filter and illustrated the value of a consider PHD filter implementation.

Section 4 then proposed and detailed an approximate technique for terrain-aided navigation using multitarget tracking. Discussion was given to illustrate why the proposed approximation was more appropriate than standard methods in literature for the problems considered by this dissertation, specific and general instructions were presented for conforming standard set-valued multitarget filters to the proposed approximation, and extensive numerical studies were performed to illuminate the key features of the approach. Overall, the proposed technique was shown to be an effective methodology for incorporating terrain-related data into a navigation solution, and it was demonstrated that the proposed technique can naturally process traditional dataypes, such as information from altimeters and inertial measurement units. Interestingly, it was discovered that multitarget filters that drastically vary in complexity, such as the PHD and δ-GLMB filters, exhibit nearly the same navigation performance, and that their differences reside in mapping performance. Therefore, a navigator can select any of the numerous multitarget filters for a terrain-aided navigation application by selecting the filter that satisfies the mapping requirements of the mission. The section continued to further propose a fusion strategy to augment, rather than replace, existing, traditional navigation strategies by "bolting on" the proposed filter to save

substantially on development costs and provide feasibility. Numerical simulation indicates that the fusion strategy produces comparable results to the methodology that processes all of the data under a common framework.

Section 4 concluded by deriving a new filter based on the PHD filter that, rather than estimating the contents of a single set, estimates a known number of different sets, each having their own dimensional, detection, and dynamical properties. An analytical, closed-form GM solution was presented, and numerical interrogations of the new filter indicated that, when compared to an analogous method in literature, map feature localization performance is improved at the expense of inheriting a cardinality bias and loss of the famous linear complexity of the PHD filter. The new filter was applied to a navigation example, where a map is partitioned into two subsets, one smaller than the other, and was demonstrated that sufficient navigation performance can be obtained by only updating a small portion of the total set, rather than the entire map. Inspired by the consider filtering of the previous section, the concept interprets a subset of the map as a collection of prioritized features and skips updates to the remaining map features entirely. This resulted in a substantial reduction of the required computational runtime and indicated the practical feasibility of the new filter for navigation applications.

Section 5 applied the proposed terrain-aided navigation architecture with decentralized fusion and square-root consider filters to a lunar landing scenario and demonstrated the architecture's flexible capabilities, such as permitting the use of inertial measurement unit-based time updates, processing multiple datatypes, and use of terrain maps based on real lunar data. These simulations contained a great deal of realism, including IMU models with non-orthogonality and misalignment terms, realistic sensor biases and noise statistics, realistic descent trajectories, and modeling of detection failures and errors due to image processing. Additionally, the scenario demonstrated the proposed architecture's capability to autonomously learn uninitialized map features using the collected images, as well as the scheme's capability to perform well for large map sizes. It was demonstrated, however, that larger map sizes somewhat hinder the architecture's ability to improve a vehicle state estimate, and so a navigator should carefully consider the map and observation geometry when designing a scheme for a given mission.

Future research directions include devising more efficient implementations of the proposed approach because, despite the computational performance advantages outlined by these results, this terrain aiding strategy still poses a significant burden to the types of flight processors that are cleared for use in modern spaceflight. The astronomical investment required for sufficient radiation-hardening and environmental testing of a new processor has forced the spaceflight industry to almost inexorably tie itself to processors that are archaic by commercial computing standards. The reality is that commercial computing resources do not possess the robustness required to operate in orbital conditions, and the flight-tested hardware that is commonly applied to these missions is, while very limited on computational resources, very reliable. Therefore, it stands to reason that new approximations, or improvements to the proposed approximation, should be developed to encourage application of the described methods.

Additionally, the results of Section 5 promote further research into initializing the filter with no initial map at all, where the vehicle is required to "start from scratch." It seems possible that this approach could offer the necessary balance between performance and computational burden, since the filter would only instantiate and track features it collects observations of without carrying along additional, unused features. This further motivates research toward a memory-loss function that could be devised for such problems because, for planetary descent and landing in particular, once a field of terrain has been passed, the vehicle is not expected to return and observe that field again. It could be fruitful to limit the filter's memory of such features and remove them from consideration once they no longer contribute to the navigation solution.

Ultimately, it is prudent that a navigation implementation prioritize the vehicle state estimate above all things. Therefore, promising directions are toward methods that reduce computational burden at the expense of mapping, rather than navigation, performance, without returning to the *ad hoc*, heuristically driven techniques that hope to shoehorn terrain camera data in to a traditional Kalman filter architecture. Ideally, the desired method is one that requires no restrictive assumptions on the observation geometry and one that does not rely heavily upon image processing techniques.

APPENDIX A.

DIAGONAL, FULL RANK UPDATE TO CHOLESKY FACTORS

Consider the update

$$\boldsymbol{S}\boldsymbol{S}^T = \boldsymbol{A}\boldsymbol{A}^T + \boldsymbol{U}\boldsymbol{U}^T,$$

where $\boldsymbol{S}$ and $\boldsymbol{A}$ are upper triangular matrices of dimension n and $\boldsymbol{U}$ is a diagonal matrix of dimension n. Note this this update can be interpreted as a full-rank update of upper triangular matrix $\boldsymbol{A}$ with $\boldsymbol{U}$ under the constraint that $\boldsymbol{U}$ is diagonal. A numerical algorithm for performing this update is given in Algorithm 8. Proving this result requires a certain deal of arithmetic gymnastics but is not inherently difficult, and so a proof is omitted.

Algorithm 8 Diagonal Update of Upper Triangular Cholesky Factor

function CHOL_DIAG_UPDATE($\boldsymbol{A}$, $\boldsymbol{U}$)
  $\boldsymbol{A}$ and $\boldsymbol{U}$ are of dimension $n \times n$
  $\boldsymbol{S} = \boldsymbol{0}_{n\times n}$
  $s_{n,n} = \sqrt{a_{n,n}^2 + u_{n,n}^2}$
  $\boldsymbol{s}_{1:n-1,n} = \boldsymbol{a}_{1:n-1,n} \cdot (a_{n,n}/s_{n,n})$
  $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{A}^T$, let $b_{i,j}$ denote the $(i,j)^{\text{th}}$ entry of $\boldsymbol{B}$
  for $i = 1, \ldots, n-1$ **do**
    for $j = 1, \ldots, i$ **do**
      if $i \neq j$ **then**
        $s_{n-i,n-j} = (b_{i,j} - \boldsymbol{s}_{n-i,n-j+1:n}\boldsymbol{s}_{n-j,n-j+1:n}^T)/s_{n-j,n-j}$
      else
        $s_{n-i,n-j} = \sqrt{b_{i,j} - \boldsymbol{s}_{n-i,n-j+1:n}\boldsymbol{s}_{n-i,n-j+1:n}^T + u_{n-i,n-j}^2}$
return $\boldsymbol{S}$

APPENDIX B.

THE METHOD OF MOMENTS FOR GAUSSIAN MIXTURES

It is often of interest to combine a number of GM components into a single mean and covariance representation, such as for state estimation or model reduction. It is well-known that the mean and covariance of an L-component GM of the form

$$p(\boldsymbol{x}) = \sum_{\ell=1}^{L} w_\ell \; p_g(\boldsymbol{x}\,;\, \boldsymbol{m}_{x,\ell}, \boldsymbol{P}_{xx,\ell})$$

can be computed using the method of moments, yielding

$$\begin{aligned}
\bar{w} &= \sum_{\ell=1}^{L} w_\ell \\
\bar{\boldsymbol{m}}_x &= \sum_{\ell=1}^{L} \frac{w_\ell}{\bar{w}} \boldsymbol{m}_{x,\ell} \\
\bar{\boldsymbol{P}}_{xx} &= \sum_{\ell=1}^{L} \frac{w_\ell}{\bar{w}} \left(\boldsymbol{P}_{xx,\ell} + \boldsymbol{m}_{x,\ell} \boldsymbol{m}_{x,\ell}^T \right) - \bar{\boldsymbol{m}}_x \bar{\boldsymbol{m}}_x^T \,,
\end{aligned}$$

where $\bar{\boldsymbol{m}}_x$ and $\bar{\boldsymbol{P}}_{xx}$ are the combined mean and covariance, respectively. For a method employing square-root factors in lieu of covariance, an expression must be obtained for $\bar{\boldsymbol{S}}_{xx,\ell}$ such that $\bar{\boldsymbol{S}}_{xx,\ell} \bar{\boldsymbol{S}}_{xx,\ell}^T = \bar{\boldsymbol{P}}_{xx,\ell}$. To that end, rewrite $\bar{\boldsymbol{P}}_{xx}$ as

$$\bar{\boldsymbol{P}}_{xx} = \sum_{\ell=1}^{L} \frac{w_\ell}{\bar{w}} \left[\boldsymbol{P}_{xx,\ell} + (\boldsymbol{m}_{x,\ell} - \bar{\boldsymbol{m}}_x)(\boldsymbol{m}_{x,\ell} - \bar{\boldsymbol{m}}_x)^T \right] \,.$$

Defining

$$\boldsymbol{d}_{x,\ell} = \boldsymbol{m}_{x,\ell} - \bar{\boldsymbol{m}}_x$$

permits this to be rewritten as

$$\bar{\boldsymbol{P}}_{xx} = \sum_{\ell=1}^{L} \frac{w_\ell}{\bar{w}} \left(\boldsymbol{P}_{xx,\ell} + \boldsymbol{d}_{x,\ell} \boldsymbol{d}_{x,\ell}^T \right) \,.$$

Expanding the sum yields

$$\bar{\boldsymbol{P}}_{xx} = \frac{1}{\bar{w}} \left[w_1 \left(\boldsymbol{P}_{xx,1} + \boldsymbol{d}_{x,1} \boldsymbol{d}_{x,1}^T \right) + \cdots + w_L \left(\boldsymbol{P}_{xx,L} + \boldsymbol{d}_{x,L} \boldsymbol{d}_{x,L}^T \right) \right] \,,$$

and, by rearranging, the expression becomes

$$\bar{\boldsymbol{P}}_{xx} = \frac{1}{\bar{w}} \left[w_1 \boldsymbol{P}_{xx,1} + \cdots + w_L \boldsymbol{P}_{xx,L} + w_1 \boldsymbol{d}_{x,1} \boldsymbol{d}_{x,1}^T + \cdots + w_L \boldsymbol{d}_{x,L} \boldsymbol{d}_{x,L}^T \right] .$$

Substituting for square-root factors yields

$$\bar{\boldsymbol{S}}_{xx} \bar{\boldsymbol{S}}_{xx}^T = \frac{1}{\bar{w}} \left[w_1 \boldsymbol{S}_{xx,1} \boldsymbol{S}_{xx,1}^T + \cdots + w_L \boldsymbol{S}_{xx,L} \boldsymbol{S}_{xx,L}^T + w_1 \boldsymbol{d}_{x,1} \boldsymbol{d}_{x,1}^T + \cdots + w_L \boldsymbol{d}_{x,L} \boldsymbol{d}_{x,L}^T \right] ,$$

and factoring terms yields

$$\bar{\boldsymbol{S}}_{xx} \bar{\boldsymbol{S}}_{xx}^T = \frac{1}{\bar{w}} \Big[\; \sqrt{w_1} \boldsymbol{S}_{xx,1} \; \Big| \; \ldots \; \Big| \; \sqrt{w_L} \boldsymbol{S}_{xx,L} \; \Big| \; \sqrt{w_1} \boldsymbol{d}_{x,1} \; \Big| \; \ldots \; \Big| \; \sqrt{w_L} \boldsymbol{d}_{x,L} \; \Big] \Big[... \Big]^T ,$$

such that it can be concluded that a form of the desired factor is

$$\frac{1}{\sqrt{\bar{w}}} \Big[\; \sqrt{w_1} \boldsymbol{S}_{xx,1} \; \Big| \; \ldots \; \Big| \; \sqrt{w_L} \boldsymbol{S}_{xx,L} \; \Big| \; \sqrt{w_1} \boldsymbol{d}_{x,1} \; \Big| \; \ldots \; \Big| \; \sqrt{w_L} \boldsymbol{d}_{x,L} \; \Big] .$$

Of course, this is non-square, and therefore the desired square-root factor is obtained with the RQ-decomposition

$$\bar{\boldsymbol{S}}_{xx} = \mathrm{rq} \left\{ \frac{1}{\sqrt{\bar{w}}} \Big[\; \sqrt{w_1} \boldsymbol{S}_{xx,1} \; \Big| \; \ldots \; \Big| \; \sqrt{w_L} \boldsymbol{S}_{xx,L} \; \Big| \; \sqrt{w_1} \boldsymbol{d}_{x,1} \; \Big| \; \ldots \; \Big| \; \sqrt{w_L} \boldsymbol{d}_{x,L} \; \Big] \right\} ,$$

which is the desired result.

For a description of how to compute the mean and covariance of a GM containing attitude states, see the appendix of [143]. A square-root form with attitude then follows immediately from the discussion above.

APPENDIX C.

MATHEMATICAL PROOFS

APPENDIX CONTENTS

C.1. PROOF OF HHR MATRIX PROPERTIES

(Referring to discussion on hyperbolic Householder reflection properties in Section 3.2.1.) Proving the claim of nonsingularity is as simple as checking the determinant of (both sides of) Eq. (3.18), yielding

$$\det \boldsymbol{\Xi Y \Xi}^T = \det\{\boldsymbol{Y}\}\,.$$

The determinant of a square matrix product is the product of determinants, such that

$$\det\{\boldsymbol{\Xi}\}\det\{\boldsymbol{Y}\}\det\{\boldsymbol{\Xi}^T\} = \det\{\boldsymbol{Y}\}\,,$$

and clearly $\det\{\boldsymbol{Y}\} = \pm 1$. Therefore,

$$\det\{\boldsymbol{\Xi}\}\det\{\boldsymbol{\Xi}^T\} = 1\,,$$

which implies that $\det\{\boldsymbol{\Xi}\} = \pm 1$, supporting the claim.

To demonstrate hyperbolic symmetry, utilizing the fact that $\boldsymbol{\Xi}$ is proven to be nonsingular and Eq. (3.18), one can obtain

$$\begin{aligned}\boldsymbol{\Xi}\boldsymbol{Y}\boldsymbol{\Xi}^T &= (\boldsymbol{Y}\boldsymbol{\Xi}^{-T}\boldsymbol{Y}^{-1})\boldsymbol{Y}(\boldsymbol{Y}^{-1}\boldsymbol{\Xi}\boldsymbol{Y}) \\ &= \boldsymbol{Y}\boldsymbol{\Xi}^{-T}\boldsymbol{Y}^{-1}\boldsymbol{\Xi}\boldsymbol{Y} \\ &= \boldsymbol{Y}\boldsymbol{\Xi}^{-T}\boldsymbol{\Xi}^T \\ &= \boldsymbol{Y} \\ &= \boldsymbol{\Xi}^T\boldsymbol{Y}\boldsymbol{\Xi}\end{aligned}$$

since $\boldsymbol{\Xi} = \boldsymbol{Y}\boldsymbol{\Xi}^{-T}\boldsymbol{Y}^{-1}$. ■

C.2. PROOF OF EQ. (3.19)

Since $\boldsymbol{v} = \boldsymbol{\Xi}^T\boldsymbol{u}$, one can left multiply by $\boldsymbol{Y}$ to obtain

$$\boldsymbol{Y}\boldsymbol{v} = \boldsymbol{Y}\boldsymbol{\Xi}^T\boldsymbol{u}\,,$$

and then again left multiply, this time by $\boldsymbol{v}^T$, yielding

$$\begin{aligned}\boldsymbol{v}^T\boldsymbol{Y}\boldsymbol{v} &= \boldsymbol{v}^T\boldsymbol{Y}\boldsymbol{\Xi}^T\boldsymbol{u} \\ &= \boldsymbol{u}^T\boldsymbol{\Xi}\boldsymbol{Y}\boldsymbol{\Xi}^T\boldsymbol{u}\,.\end{aligned}$$

Then, simply noting that $\boldsymbol{\Xi}$ is hypernormal with $\boldsymbol{Y}$ such that $\boldsymbol{Y} = \boldsymbol{\Xi}\boldsymbol{Y}\boldsymbol{\Xi}^T$ produces the claimed result. ■

C.3. PROOF OF EQ. (3.28)

The prediction of the augmented covariance can be written in terms of its square-root factor as

$$\bar{\boldsymbol{S}}_k^-(\bar{\boldsymbol{S}}_k^-)^T = \sum_{\ell=1}^{q_{k-1}^+} w_{c,k-1}^{+(\ell)}(\bar{\boldsymbol{x}}_k^{-(\ell)} - \bar{\boldsymbol{m}}_k^-)(\bar{\boldsymbol{x}}_k^{-(\ell)} - \bar{\boldsymbol{m}}_k^-)^T\,,$$

and expanding the sums out to account for the r negative weights (recalling that the first r are negative) yields

$$\begin{aligned}\bar{\boldsymbol{S}}_k^-(\bar{\boldsymbol{S}}_k^-)^T = &-|w_{c,k-1}^{+(1)}|(\bar{\boldsymbol{x}}_k^{-(1)} - \bar{\boldsymbol{m}}_k^-)(\bar{\boldsymbol{x}}_k^{-(1)} - \bar{\boldsymbol{m}}_k^-)^T \\ &\quad - \cdots - |w_{c,k-1}^{+(r)}|(\bar{\boldsymbol{x}}_k^{-(r)} - \bar{\boldsymbol{m}}_k^-)(\bar{\boldsymbol{x}}_k^{-(r)} - \bar{\boldsymbol{m}}_k^-)^T \\ &+ |w_{c,k-1}^{+(r+1)}|(\bar{\boldsymbol{x}}_k^{-(r+1)} - \bar{\boldsymbol{m}}_k^-)(\bar{\boldsymbol{x}}_k^{-(r+1)} - \bar{\boldsymbol{m}}_k^-)^T \\ &\quad + \cdots + |w_{c,k-1}^{+(q_{k-1}^+)}|(\bar{\boldsymbol{x}}_k^{-(q_{k-1}^+)} - \bar{\boldsymbol{m}}_k^-)(\bar{\boldsymbol{x}}_k^{-(q_{k-1}^+)} - \bar{\boldsymbol{m}}_k^-)^T\,.\end{aligned}$$

Collecting terms, defining a signature matrix, and using the notation defined previously, this can be rewritten as

$$\bar{\boldsymbol{S}}_k^-(\bar{\boldsymbol{S}}_k^-)^T = \left[\ (\bar{\boldsymbol{x}}_k^{-(1:r)} \ominus \bar{\boldsymbol{m}}_k^-)\boldsymbol{W}^{(1:r)} \ \middle|\ (\bar{\boldsymbol{x}}_k^{-(r+1:q_{k-1}^+)} \ominus \bar{\boldsymbol{m}}_k^-)\boldsymbol{W}^{(r+1:q_{k-1}^+)}\ \right] \boldsymbol{Y} \left[\cdots\right]^T$$

where $[\cdots]^T$ is intended to mean the same argument as on the left but transposed and the signature matrix is given as

$$\boldsymbol{Y} = \text{blkdiag}\{-\boldsymbol{I}_{r\times r}, \boldsymbol{I}_{(q_{k-1}^+ - r)\times(q_{k-1}^+ - r)}\}\,.$$

Since the pseudo-orthogonal hyperbolic Householder reflection matrix $\boldsymbol{\Xi}$ obeys the properties that $\boldsymbol{\Xi}\boldsymbol{Y}\boldsymbol{\Xi}^T = \boldsymbol{Y}$ and

$$\left[\begin{array}{c|c} (\bar{\boldsymbol{x}}_k^{-(1:r)} \ominus \bar{\boldsymbol{m}}_k^-)\boldsymbol{W}^{(1:r)} & (\bar{\boldsymbol{x}}_k^{-(r+1:q_{k-1}^+)} \ominus \bar{\boldsymbol{m}}_k^-)\boldsymbol{W}^{(r+1:q_{k-1}^+)} \end{array}\right] \boldsymbol{\Xi} = \left[\begin{array}{c|c} \boldsymbol{0}_{\bar{n}\times(q_{k-1}^+-\bar{n})} & \bar{\boldsymbol{S}}^\dagger \end{array}\right]$$

(i.e. that it produces an upper triangular matrix $\bar{\boldsymbol{S}}^\dagger$), it can be concluded that $\bar{\boldsymbol{S}}^\dagger = \bar{\boldsymbol{S}}_k^-$ and the claimed result is supported. ■

C.4. PROOF OF EQS. (3.32)

Start by manipulating and multiplying out Eq. (3.31) on pp. 92 to yield

$$\bar{\boldsymbol{S}}_k^*(\bar{\boldsymbol{S}}_k^*)^T = \begin{bmatrix} \boldsymbol{F}_{x,k-1}\boldsymbol{S}_{xx,k-1}^+ & \boldsymbol{F}_{x,k-1}\boldsymbol{S}_{xc,k-1}^+ + \boldsymbol{F}_{c,k-1}\boldsymbol{S}_{cc,k-1}^+ & \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1} & \boldsymbol{0}_{n_x\times n_c} \\ \boldsymbol{0}_{n_c\times n_x} & \boldsymbol{S}_{cc,k-1}^+ & \boldsymbol{0}_{n_c\times n_x} & \boldsymbol{0}_{n_c\times n_c} \end{bmatrix} \begin{bmatrix} \ldots \end{bmatrix}^T .$$

Expanding further yields

$$\bar{\boldsymbol{S}}_k^*(\bar{\boldsymbol{S}}_k^*)^T = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_2 \\ \boldsymbol{A}_2^T & \boldsymbol{A}_3 \end{bmatrix},$$

where

$$\begin{aligned} \boldsymbol{A}_1 &= \boldsymbol{F}_{x,k-1}\boldsymbol{S}_{xx,k-1}^+(\boldsymbol{S}_{xx,k-1}^+)^T\boldsymbol{F}_{x,k-1}^T + \boldsymbol{F}_{x,k-1}\boldsymbol{S}_{xc,k-1}^+(\boldsymbol{S}_{xc,k-1}^+)^T\boldsymbol{F}_{x,k-1}^T \\ &\quad + \boldsymbol{F}_{x,k-1}\boldsymbol{S}_{xc,k-1}^+(\boldsymbol{S}_{cc,k-1}^+)^T\boldsymbol{F}_{c,k-1}^T + \boldsymbol{F}_{c,k-1}\boldsymbol{S}_{cc,k-1}^+(\boldsymbol{S}_{xc,k-1}^+)^T\boldsymbol{F}_{x,k-1}^T \\ &\quad + \boldsymbol{F}_{c,k-1}\boldsymbol{S}_{cc,k-1}^+(\boldsymbol{S}_{cc,k-1}^+)^T\boldsymbol{F}_{c,k-1}^T + \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1}(\boldsymbol{S}_{ww,k-1})^T\boldsymbol{F}_{w,k-1}^T \\ \boldsymbol{A}_2 &= [\boldsymbol{F}_{x,k-1}\boldsymbol{S}_{xc,k-1}^+ + \boldsymbol{F}_{c,k-1}\boldsymbol{S}_{cc,k-1}^+](\boldsymbol{S}_{cc,k-1}^+)^T \\ \boldsymbol{A}_3 &= \boldsymbol{S}_{cc,k-1}^+(\boldsymbol{S}_{cc,k-1}^+)^T, \end{aligned}$$

Note that

$$\begin{aligned} \bar{\boldsymbol{S}}_k^*(\bar{\boldsymbol{S}}_k^*)^T &= \begin{bmatrix} \boldsymbol{S}_{xx,k}^* & \boldsymbol{S}_{xc,k}^* \\ \boldsymbol{0}_{n_c\times n_x} & \boldsymbol{S}_{cc,k}^* \end{bmatrix} \begin{bmatrix} (\boldsymbol{S}_{xx,k}^*)^T & \boldsymbol{0}_{n_x\times n_c} \\ (\boldsymbol{S}_{xc,k}^*)^T & (\boldsymbol{S}_{cc,k}^*)^T \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{S}_{xx,k}^*(\boldsymbol{S}_{xx,k}^*)^T + \boldsymbol{S}_{xc,k}^*(\boldsymbol{S}_{xc,k}^*)^T & \boldsymbol{S}_{xc,k}^*(\boldsymbol{S}_{cc,k}^*)^T \\ \boldsymbol{S}_{cc,k}^*(\boldsymbol{S}_{xc,k}^*)^T & \boldsymbol{S}_{cc,k}^*(\boldsymbol{S}_{cc,k}^*)^T \end{bmatrix}, \end{aligned}$$

and therefore

$$\begin{aligned}
\boldsymbol{A}_1 &= \boldsymbol{S}^*_{xx,k}(\boldsymbol{S}^*_{xx,k})^T + \boldsymbol{S}^*_{xc,k}(\boldsymbol{S}^*_{xc,k})^T \\
\boldsymbol{A}_2 &= \boldsymbol{S}^*_{xc,k}(\boldsymbol{S}^*_{cc,k})^T \\
\boldsymbol{A}_3 &= \boldsymbol{S}^*_{cc,k}(\boldsymbol{S}^*_{cc,k})^T .
\end{aligned}$$

First, to prove Eq. (3.32b), equate like terms for $\boldsymbol{A}_2$ to obtain

$$\boldsymbol{S}^*_{xc,k}(\boldsymbol{S}^*_{cc,k})^T = [\boldsymbol{F}_{x,k-1}\boldsymbol{S}^+_{xc,k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{S}^+_{cc,k-1}](\boldsymbol{S}^+_{cc,k-1})^T$$

and solving, since $\boldsymbol{S}^*_{cc,k}$ is always invertible, shows

$$\boldsymbol{S}^*_{xc,k} = \boldsymbol{F}_{x,k-1}\boldsymbol{S}^+_{xc,k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{S}^+_{cc,k-1} ,$$

proving the claim.

Then, to prove Eq. (3.32a), equate like terms for $\boldsymbol{A}_1$ and isolate the $\boldsymbol{S}^*_{xx,k}$ term to obtain

$$\begin{aligned}
\boldsymbol{S}^*_{xx,k}(\boldsymbol{S}^*_{xx,k})^T = {} & \boldsymbol{F}_{x,k-1}\boldsymbol{S}^+_{xx,k-1}(\boldsymbol{S}^+_{xx,k-1})^T\boldsymbol{F}^T_{x,k-1} + \boldsymbol{F}_{x,k-1}\boldsymbol{S}^+_{xc,k-1}(\boldsymbol{S}^+_{xc,k-1})^T\boldsymbol{F}^T_{x,k-1} \\
& + \boldsymbol{F}_{x,k-1}\boldsymbol{S}^+_{xc,k-1}(\boldsymbol{S}^+_{cc,k-1})^T\boldsymbol{F}^T_{c,k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{S}^+_{cc,k-1}(\boldsymbol{S}^+_{xc,k-1})^T\boldsymbol{F}^T_{x,k-1} \\
& + \boldsymbol{F}_{c,k-1}\boldsymbol{S}^+_{cc,k-1}(\boldsymbol{S}^+_{cc,k-1})^T\boldsymbol{F}^T_{c,k-1} + \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1}(\boldsymbol{S}_{ww,k-1})^T\boldsymbol{F}^T_{w,k-1} \\
& - \boldsymbol{S}^*_{xc,k}(\boldsymbol{S}^*_{xc,k})^T .
\end{aligned}$$

Substituting for $\boldsymbol{S}^*_{xc,k}$ shows that

$$\begin{aligned}\boldsymbol{S}^*_{xx,k}(\boldsymbol{S}^*_{xx,k})^T = {} & \boldsymbol{F}_{x,k-1}\boldsymbol{S}^+_{xx,k-1}(\boldsymbol{S}^+_{xx,k-1})^T\boldsymbol{F}^T_{x,k-1} + \boldsymbol{F}_{x,k-1}\boldsymbol{S}^+_{xc,k-1}(\boldsymbol{S}^+_{xc,k-1})^T\boldsymbol{F}^T_{x,k-1} \\ & + \boldsymbol{F}_{x,k-1}\boldsymbol{S}^+_{xc,k-1}(\boldsymbol{S}^+_{cc,k-1})^T\boldsymbol{F}^T_{c,k-1} + \boldsymbol{F}_{c,k-1}\boldsymbol{S}^+_{cc,k-1}(\boldsymbol{S}^+_{xc,k-1})^T\boldsymbol{F}^T_{x,k-1} \\ & + \boldsymbol{F}_{c,k-1}\boldsymbol{S}^+_{cc,k-1}(\boldsymbol{S}^+_{cc,k-1})^T\boldsymbol{F}^T_{c,k-1} + \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1}(\boldsymbol{S}_{ww,k-1})^T\boldsymbol{F}^T_{w,k-1} \\ & - \boldsymbol{F}_{x,k-1}\boldsymbol{S}^+_{xc,k-1}(\boldsymbol{S}^+_{xc,k-1})^T\boldsymbol{F}^T_{x,k-1} - \boldsymbol{F}_{c,k-1}\boldsymbol{S}^+_{cc,k-1}(\boldsymbol{S}^+_{cc,k-1})^T\boldsymbol{F}^T_{c,k-1} \\ & - \boldsymbol{F}_{x,k-1}\boldsymbol{S}^+_{xc,k-1}(\boldsymbol{S}^+_{cc,k-1})^T\boldsymbol{F}^T_{c,k-1} - \boldsymbol{F}_{c,k-1}\boldsymbol{S}^+_{cc,k-1}(\boldsymbol{S}^+_{xc,k-1})^T\boldsymbol{F}^T_{x,k-1} \\ = {} & \boldsymbol{F}_{x,k-1}\boldsymbol{S}^+_{xx,k-1}(\boldsymbol{S}^+_{xx,k-1})^T\boldsymbol{F}^T_{x,k-1} + \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1}(\boldsymbol{S}_{ww,k-1})^T\boldsymbol{F}^T_{w,k-1}\,,\end{aligned}$$

and, after some factoring, it can be concluded that

$$\boldsymbol{S}^*_{xx,k} = \mathrm{rq}\left\{\left[\begin{array}{c|c} \boldsymbol{F}_{x,k-1}\boldsymbol{S}^+_{xx,k-1} & \boldsymbol{F}_{w,k-1}\boldsymbol{S}_{ww,k-1}\end{array}\right]\right\}\,,$$

supporting the claim.

The result of Eq. (3.32c) is trivially supported by inspection. ■

C.5. PROOF OF LEMMA (3.3)

Start with an integral of the form

$$I_1 = \int p_g\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{b}\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{F} & \boldsymbol{G}\\ \boldsymbol{0} & \boldsymbol{M}\end{bmatrix}\begin{bmatrix}\boldsymbol{\xi}\\ \boldsymbol{b}\end{bmatrix},\begin{bmatrix}\boldsymbol{Q} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{\Omega}\end{bmatrix}\right)\times p_g\left(\begin{bmatrix}\boldsymbol{\xi}\\ \boldsymbol{b}\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{m}\\ \boldsymbol{p}\end{bmatrix},\begin{bmatrix}\boldsymbol{P} & \boldsymbol{L}\\ \boldsymbol{L}^T & \boldsymbol{B}\end{bmatrix}\right)\mathrm{d}\begin{bmatrix}\boldsymbol{\xi}\\ \boldsymbol{b}\end{bmatrix}\,.$$

Noting that I_1 is of the form of Lemma 3.1, an application of this identity yields

$$I_1 = p_g\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{b}\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{F} & \boldsymbol{G}\\ \boldsymbol{0} & \boldsymbol{M}\end{bmatrix}\begin{bmatrix}\boldsymbol{m}\\ \boldsymbol{p}\end{bmatrix},\begin{bmatrix}\boldsymbol{F} & \boldsymbol{G}\\ \boldsymbol{0} & \boldsymbol{M}\end{bmatrix}\begin{bmatrix}\boldsymbol{P} & \boldsymbol{L}\\ \boldsymbol{L}^T & \boldsymbol{B}\end{bmatrix}\begin{bmatrix}\boldsymbol{F}^T & \boldsymbol{0}\\ \boldsymbol{G}^T & \boldsymbol{M}^T\end{bmatrix}+\begin{bmatrix}\boldsymbol{Q} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{\Omega}\end{bmatrix}\right)\,,$$

which eliminates the integral from the expression and "condenses" I_1 into a single, joint Gaussian distribution over $\boldsymbol{x}$ and $\boldsymbol{b}$. Carrying out multiplication of mean and covariance terms within the Gaussian yields

$$I_1 = p_g\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{b}\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{F}\boldsymbol{m}+\boldsymbol{G}\boldsymbol{p}\\ \boldsymbol{M}\boldsymbol{p}\end{bmatrix},\begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{F}\boldsymbol{L}\boldsymbol{M}^T+\boldsymbol{G}\boldsymbol{B}\boldsymbol{M}^T\\ \boldsymbol{M}\boldsymbol{L}^T\boldsymbol{F}^T+\boldsymbol{M}\boldsymbol{B}\boldsymbol{G}^T & \boldsymbol{M}\boldsymbol{B}\boldsymbol{M}^T+\boldsymbol{\Omega}\end{bmatrix}\right)\,,$$

with

$$\boldsymbol{\Sigma} = \boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^T + \boldsymbol{G}\boldsymbol{B}\boldsymbol{G}^T + \boldsymbol{G}\boldsymbol{L}^T\boldsymbol{F}^T + \boldsymbol{F}\boldsymbol{L}\boldsymbol{G}^T + \boldsymbol{Q}\,.$$

Defining terms gives a simplified expression as

$$I_1 = p_g\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{b}\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\eta}\end{bmatrix},\begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{\Pi}\\ \boldsymbol{\Pi}^T & \boldsymbol{\Xi}\end{bmatrix}\right),$$

where

$$\begin{aligned}
\boldsymbol{\mu} &= \boldsymbol{F}\boldsymbol{m} + \boldsymbol{G}\boldsymbol{p}\\
\boldsymbol{\eta} &= \boldsymbol{M}\boldsymbol{p}\\
\boldsymbol{\Sigma} &= \boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^T + \boldsymbol{G}\boldsymbol{B}\boldsymbol{G}^T + \boldsymbol{G}\boldsymbol{L}^T\boldsymbol{F}^T + \boldsymbol{F}\boldsymbol{L}\boldsymbol{G}^T + \boldsymbol{Q}\\
\boldsymbol{\Pi} &= \boldsymbol{F}\boldsymbol{L}\boldsymbol{M}^T + \boldsymbol{G}\boldsymbol{B}\boldsymbol{M}^T\\
\boldsymbol{\Xi} &= \boldsymbol{M}\boldsymbol{B}\boldsymbol{M}^T + \boldsymbol{\Omega}\,,
\end{aligned}$$

yielding the claimed result. ■

C.6. PROOF OF LEMMA (3.4)

Start with an integral of the form

$$I_2 = \int p_g(\boldsymbol{z}\,;\,\boldsymbol{H}\boldsymbol{x} + \boldsymbol{J}\boldsymbol{b}, \boldsymbol{R}) p_g\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{b}\end{bmatrix}\,;\,\begin{bmatrix}\boldsymbol{m}\\ \boldsymbol{p}\end{bmatrix},\begin{bmatrix}\boldsymbol{P} & \boldsymbol{L}\\ \boldsymbol{L}^T & \boldsymbol{B}\end{bmatrix}\right)\mathrm{d}\boldsymbol{b}\,.$$

Rewriting the joint Gaussian over $\boldsymbol{x}$ and $\boldsymbol{b}$ as the product of their marginal Gaussians using the properties of jointly Gaussian random variables such that [34]

$$I_2 = \int p_g\left(\boldsymbol{z}\,;\,\boldsymbol{H}\boldsymbol{x} + \boldsymbol{J}\boldsymbol{b}, \boldsymbol{R}\right) p_g\left(\boldsymbol{x}\,;\,\boldsymbol{m}, \boldsymbol{P}\right) p_g\left(\boldsymbol{b}\,;\,\boldsymbol{p} + \boldsymbol{L}^T\boldsymbol{P}^{-1}(\boldsymbol{x} - \boldsymbol{m}), \boldsymbol{B} - \boldsymbol{L}^T\boldsymbol{P}^{-1}\boldsymbol{L}\right)\mathrm{d}\boldsymbol{b}\,.$$

Pull the Gaussian which does not functionally depend on $\boldsymbol{b}$ out of the integral, yielding

$$I_2 = p_g\left(\boldsymbol{x}\,;\,\boldsymbol{m},\boldsymbol{P}\right)\int p_g\left(\boldsymbol{z}\,;\,\boldsymbol{H}\boldsymbol{x}+\boldsymbol{J}\boldsymbol{b},\boldsymbol{R}\right)p_g\left(\boldsymbol{b}\,;\,\boldsymbol{p}+\boldsymbol{L}^T\boldsymbol{P}^{-1}(\boldsymbol{x}-\boldsymbol{m}),\boldsymbol{B}-\boldsymbol{L}^T\boldsymbol{P}^{-1}\boldsymbol{L}\right)\mathrm{d}\boldsymbol{b}\,,$$

and use Lemma 3.1 (on the integral of the product of two Gaussians) to obtain

$$I_2 = p_g\left(\boldsymbol{x}\,;\,\boldsymbol{m},\boldsymbol{P}\right)p_g\left(\boldsymbol{z}\,;\,\boldsymbol{H}\boldsymbol{x}+\boldsymbol{J}[\boldsymbol{p}+\boldsymbol{L}^T\boldsymbol{P}^{-1}(\boldsymbol{x}-\boldsymbol{m})],\boldsymbol{J}(\boldsymbol{B}-\boldsymbol{L}^T\boldsymbol{P}^{-1}\boldsymbol{L})\boldsymbol{J}^T+\boldsymbol{R}\right)\,.$$

Then, an application of Lemma 3.2 yields

$$I_2 = q(\boldsymbol{z})p_g\left(\boldsymbol{x}\,;\,\boldsymbol{\mu},\boldsymbol{\Sigma}\right)$$

where

$$\begin{aligned}
q(\boldsymbol{z}) &= p_g\left(\boldsymbol{z}\,;\,\boldsymbol{H}\boldsymbol{m}+\boldsymbol{J}\boldsymbol{p},\boldsymbol{W}\right)\\
\boldsymbol{\mu} &= \boldsymbol{m}+\boldsymbol{K}(\boldsymbol{z}-\boldsymbol{H}\boldsymbol{m}-\boldsymbol{J}\boldsymbol{p})\\
\boldsymbol{\Sigma} &= (\boldsymbol{I}-\boldsymbol{K}\boldsymbol{H})\boldsymbol{P}-\boldsymbol{K}\boldsymbol{J}\boldsymbol{L}^T\\
\boldsymbol{K} &= \boldsymbol{C}\boldsymbol{W}^{-1}\\
\boldsymbol{C} &= \boldsymbol{P}\boldsymbol{H}^T+\boldsymbol{L}\boldsymbol{J}^T\\
\boldsymbol{W} &= \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^T+\boldsymbol{J}\boldsymbol{B}\boldsymbol{J}^T+\boldsymbol{H}\boldsymbol{L}\boldsymbol{J}^T+\boldsymbol{J}\boldsymbol{L}^T\boldsymbol{H}^T+\boldsymbol{R}\,,
\end{aligned}$$

establishing the claimed result. ■

C.7. PROOF OF EQS. (3.48)

First, substitute Eqs. (3.44) and (3.46) into the marginalized Bayes' rule of Eq. (3.41) to obtain

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k}) &= \int \frac{1}{\eta_k} g(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{c}_k) p(\boldsymbol{x}_k, \boldsymbol{c}_k|\boldsymbol{Z}_{1:k-1}) \mathrm{d}\boldsymbol{c}_k \\ &= \int \frac{1}{\eta_k} p_g(\boldsymbol{z}_k \,;\, \boldsymbol{H}_{x,k}\boldsymbol{x}_k + \boldsymbol{H}_{c,k}\boldsymbol{c}_k, \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}_{v,k}^T) \\ &\qquad \times p_g\left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{c}_k \end{bmatrix} \,;\, \begin{bmatrix} \boldsymbol{m}_{x,k}^- \\ \boldsymbol{m}_{c,k}^- \end{bmatrix}, \begin{bmatrix} \boldsymbol{P}_{xx,k}^- & \boldsymbol{P}_{xc,k}^- \\ (\boldsymbol{P}_{xc,k}^-)^T & \boldsymbol{P}_{cc,k}^- \end{bmatrix} \right) \mathrm{d}\boldsymbol{c}_k \,. \end{aligned}$$

Applying Lemma 3.4 to this expression yields

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k}) = \frac{1}{\eta_k} q(\boldsymbol{z}_k) p_g(\boldsymbol{x}_k \,;\, \boldsymbol{m}_{x,k}^+, \boldsymbol{P}_{xx,k}^+) \,,$$

where

$$q(\boldsymbol{z}_k) = p_g(\boldsymbol{z}_k \,;\, \boldsymbol{H}_{x,k}\boldsymbol{m}_{x,k}^- + \boldsymbol{H}_{c,k}\boldsymbol{m}_{c,k}^-, \boldsymbol{P}_{zz,k})$$

and the remaining terms are as defined in Eqs. (3.48). Now, note that the appropriate normalization constant takes the form

$$\begin{aligned} \eta_k = \iint & p_g(\boldsymbol{z}_k \,;\, \boldsymbol{H}_{x,k}\boldsymbol{x}_k + \boldsymbol{H}_{c,k}\boldsymbol{c}_k, \boldsymbol{H}_{v,k}\boldsymbol{P}_{vv,k}\boldsymbol{H}_{v,k}^T) \\ & \times p_g\left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{c}_k \end{bmatrix} \,;\, \begin{bmatrix} \boldsymbol{m}_{x,k}^- \\ \boldsymbol{m}_{c,k}^- \end{bmatrix}, \begin{bmatrix} \boldsymbol{P}_{xx,k}^- & \boldsymbol{P}_{xc,k}^- \\ (\boldsymbol{P}_{xc,k}^-)^T & \boldsymbol{P}_{cc,k}^- \end{bmatrix} \right) \mathrm{d}\boldsymbol{c}_k \mathrm{d}\boldsymbol{x}_k \,, \end{aligned}$$

which, via an application of Lemma 3.4 for the integral over $\boldsymbol{c}_k$, can be written as

$$\eta_k = q(\boldsymbol{z}_k) \int p_g(\boldsymbol{x}_k \,;\, \boldsymbol{m}_{x,k}^+, \boldsymbol{P}_{xx,k}^+) \mathrm{d}\boldsymbol{x}_k \,.$$

The integral compels the Gaussian density to unity, and the sole remaining term is given as

$$\eta_k = q(\boldsymbol{z}_k) \,.$$

Therefore, the posterior density is given simply by

$$p(\boldsymbol{x}_k|\boldsymbol{Z}_{1:k}) = p_g(\boldsymbol{x}_k\,;\,\boldsymbol{m}^+_{x,k},\boldsymbol{P}^+_{xx,k})\,,$$

as claimed. ■

C.8. PROOF OF EQ. (4.15)

The PGFL of the n^{th} set's surviving targets is given, by definition, as

$$G^{(n)}_{\boldsymbol{\Xi}_{k|k-1,i}}[h] = \int h^{\boldsymbol{X}^{(n)}_k} f^{(n)}(\boldsymbol{X}^{(n)}_k|\boldsymbol{X}^{(n)}_{k-1})\delta\boldsymbol{X}^{(n)}_k$$

and expanding the set integral yields

$$G^{(n)}_{\boldsymbol{\Xi}_{k|k-1,i}}[h] = f^{(n)}(\emptyset) + \sum_{j=1}^{\infty}\frac{1}{j!}\int h^{\{\boldsymbol{x}^{(n)}_{k,1},\ldots,\boldsymbol{x}^{(n)}_{k,j}\}} f^{(n)}(\{\boldsymbol{x}^{(n)}_{k,1},\ldots,\boldsymbol{x}^{(n)}_{k,j}\}|\boldsymbol{X}^{(n)}_{k-1})\mathrm{d}\boldsymbol{x}^{(n)}_{k,1}\cdots\mathrm{d}\boldsymbol{x}^{(n)}_{k,j}\,.$$

Since $\boldsymbol{\Xi}^{(n)}_{k|k-1,i}$ can only realize values $\{\emptyset\}$ or $\{\boldsymbol{x}^{(n)}_{k,i}\}$, this integral reduces substantially to

$$G^{(n)}_{\boldsymbol{\Xi}_{k|k-1,i}}[h] = f^{(n)}(\emptyset) + \int h(\boldsymbol{x}^{(n)}_{k,i}) f^{(n)}(\{\boldsymbol{x}^{(n)}_{k,i}\}|\boldsymbol{X}^{(n)}_{k-1})\mathrm{d}\boldsymbol{x}^{(n)}_{k,i}\,.$$

Finally, noting that $\boldsymbol{\Xi}^{(n)}_{k|k-1,i} = \{\boldsymbol{x}^{(n)}_{k,i}\}$ with probability $p^{(n)}_{S,k}(\boldsymbol{x}^{(n)}_{k-1,i})$ according to Markov transition kernel $f^{(n)}(\boldsymbol{x}^{(n)}_{k,i}|\boldsymbol{x}^{(n)}_{k-1,i})$ or is equal to $\{\emptyset\}$ otherwise, this expression becomes

$$G^{(n)}_{\boldsymbol{\Xi}_{k|k-1,i}}[h] = \underbrace{1 - p^{(n)}_{S,k}(\boldsymbol{x}^{(n)}_{k-1,i})}_{\boldsymbol{\Xi}^{(n)}_{k|k-1,i}=\{\emptyset\}} + \underbrace{p^{(n)}_{S,k}(\boldsymbol{x}^{(n)}_{k-1,i})\int h\cdot f^{(n)}(\boldsymbol{x}^{(n)}_{k,i}|\boldsymbol{x}^{(n)}_{k-1,i})\mathrm{d}\boldsymbol{x}^{(n)}_{k,i}}_{\boldsymbol{\Xi}^{(n)}_{k|k-1,i}=\{\boldsymbol{x}^{(n)}_{k,i}\}}\,,$$

the claimed result. ■

C.9. PROOF OF EQ. (4.25)

The PGFL $G_{k|k-1}[h_1,\ldots,h_N]$ in Eq. (4.17) governs the temporal evolution of the simultaneous motion of all N sets, and let $G_{k|k-1}[h_1,\ldots,h_N|\boldsymbol{Z}_{1:k-1}]$ denote the PGFL of the joint prior multitarget density. Recall that the PHD product of N sets can be obtained

from the joint PGFL using set derivatives, and therefore the joint *a priori* PHD can be written as

$$\prod_{n=1}^{N} v_k^-(\boldsymbol{x}_k^{(n)}) = \frac{\delta G_{k|k-1}}{\delta \boldsymbol{x}_k^{(1)} \cdots \delta \boldsymbol{x}_k^{(N)}}[h_1, \ldots, h_N | \boldsymbol{Z}_{1:k-1}]\Big|_{h_1=\cdots=h_N=1} .$$

To take these derivatives, a form must first be obtained for the joint PGFL of the *a priori* joint density, which is, by definition, given by (employing the abbreviations defined in Section 4.5.1)

$$G_{k|k-1}[h_{1:N}|\boldsymbol{Z}_{1:k-1}] = \int h^{\boldsymbol{X}_k^{(1:N)}} \pi(\boldsymbol{X}_k^{(1:N)}|\boldsymbol{Z}_{1:k-1}) \delta \boldsymbol{X}_k^{(1:N)} .$$

Recall that the (multitarget) Chapman-Kolmogorov equation states that

$$\pi(\boldsymbol{X}_k^{(1:N)}|\boldsymbol{Z}_{1:k-1}) = \int \cdots \int f(\boldsymbol{X}_k^{(1:N)}|\boldsymbol{X}_{k-1}^{(1:N)}) \pi(\boldsymbol{X}_{k-1}^{(1:N)}|\boldsymbol{Z}_{1:k-1}) \delta \boldsymbol{X}_{k-1}^{(1:N)} ,$$

and, since target transitions and densities are independent, this can be written as the set integral-product

$$\pi(\boldsymbol{X}_k^{(1:N)}|\boldsymbol{Z}_{1:k-1}) = \prod_{n=1}^{N} \int f(\boldsymbol{X}_k^{(n)}|\boldsymbol{X}_{k-1}^{(n)}) \pi(\boldsymbol{X}_{k-1}^{(n)}|\boldsymbol{Z}_{1:k-1}) \delta \boldsymbol{X}_{k-1}^{(n)} .$$

Then, substituting this into the expression for the PGFL $G_{k|k-1}[h_1, \ldots, h_N|\boldsymbol{Z}_{1:k-1}]$ yields

$$G_{k|k-1}[h_{1:N}|\boldsymbol{Z}_{1:k-1}] = \int h^{\boldsymbol{X}_k^{(1:N)}} \left[\prod_{n=1}^{N} \int f(\boldsymbol{X}_k^{(n)}|\boldsymbol{X}_{k-1}^{(n)}) \pi(\boldsymbol{X}_{k-1}^{(n)}|\boldsymbol{Z}_{1:k-1}) \delta \boldsymbol{X}_{k-1}^{(n)} \right] \delta \boldsymbol{X}_k^{(1:N)} ,$$

or, equivalently,

$$G_{k|k-1}[h_{1:N}|\boldsymbol{Z}_{1:k-1}] = \prod_{n=1}^{N} \iint h_n^{\boldsymbol{X}_k^{(n)}} f(\boldsymbol{X}_k^{(n)}|\boldsymbol{X}_{k-1}^{(n)}) \pi(\boldsymbol{X}_{k-1}^{(n)}|\boldsymbol{Z}_{1:k-1}) \delta \boldsymbol{X}_{k-1}^{(n)} \delta \boldsymbol{X}_k^{(n)} .$$

Since

$$G_{k|k-1}^{(n)}[h_n|\boldsymbol{X}_k^{(n)}] = \int h_n^{\boldsymbol{X}_k^{(n)}} f(\boldsymbol{X}_k^{(n)}|\boldsymbol{X}_{k-1}^{(n)}) \delta \boldsymbol{X}_k^{(n)}$$

this can be written as

$$G_{k|k-1}[h_{1:N}|\boldsymbol{Z}_{1:k-1}] = \prod_{n=1}^{N} \int G_{k|k-1}^{(n)}[h_n|\boldsymbol{X}_k^{(n)}]\pi(\boldsymbol{X}_{k-1}^{(n)}|\boldsymbol{Z}_{1:k-1})\delta\boldsymbol{X}_{k-1}^{(n)}\,.$$

Finally, since, by definition,

$$G_{k-1|k-1}^{(n)}[h_n|\boldsymbol{Z}_{1:k-1}] = \int h_n^{\boldsymbol{X}_{k-1}^{(n)}}\pi(\boldsymbol{X}_{k-1}^{(n)}|\boldsymbol{Z}_{1:k-1})\delta\boldsymbol{X}_{k-1}^{(n)}$$

and using Eq. (4.16), the PGFL of the *a priori* joint density is given by

$$G_{k|k-1}[h_{1:N}|\boldsymbol{Z}_{1:k-1}] = \prod_{n=1}^{N} G_{k-1|k-1}^{(n)}\left[\left(q_{S,k}^{(n)} + p_{S,k}^{(n)} f_{S,k|k-1}^{(n)}[h]\right)\left(G_{\beta}^{(n)}[h|\cdot]\right)\right] G_b^{(n)}[h_n]\,.$$

Then, the required derivative is given as

$$\prod_{n=1}^{N} v_k^{-}(\boldsymbol{x}_k^{(n)}) = \frac{\delta}{\delta\boldsymbol{x}_k^{(1)}\cdots\delta\boldsymbol{x}_k^{(N)}}\left\{\prod_{n=1}^{N} G_{k-1|k-1}^{(n)}\left[\left(q_{S,k}^{(n)} + p_{S,k}^{(n)} f_{S,k|k-1}^{(n)}[h]\right)\left(G_{\beta}^{(n)}[h|\cdot]\right)\right] G_b^{(n)}[h_n]\right\}_{h_1=\cdots=h_N=1}.$$

Since

$$\frac{\delta G_{k-1|k-1}^{(i)}}{\delta\boldsymbol{x}_k^{(j)}}[h_n|\boldsymbol{Z}_{1:k-1}] = 0 \qquad \forall\ i \neq j$$

this becomes

$$\prod_{n=1}^{N} v_k^{-}(\boldsymbol{x}_k^{(n)}) = \prod_{n=1}^{N} \frac{\delta}{\delta\boldsymbol{x}_k^{(n)}}\left\{G_{k-1|k-1}^{(n)}\left[\left(q_{S,k}^{(n)} + p_{S,k}^{(n)} f_{S,k|k-1}^{(n)}[h]\right)\left(G_{\beta}^{(n)}[h|\cdot]\right)\right] G_b^{(n)}[h_n]\right\}_{h_1=\cdots=h_N=1}.$$

Taking this derivative[1] and then setting $h_1 = \cdots = h_N = 1$ yields

$$\prod_{n=1}^{N} v_k^-(\boldsymbol{x}_k^{(n)}) = \prod_{n=1}^{N} \Bigg[\int p_{S,k}^{(n)}(\boldsymbol{x}_k^{(n)}) f^{(n)}(\boldsymbol{x}_k^{(n)}|\boldsymbol{x}_{k-1}^{(n)}) v_{k-1}^{+(n)}(\boldsymbol{x}_{k-1}^{(n)}) \mathrm{d}\boldsymbol{x}_{k-1}^{(n)} + \int \beta^{(n)}(\boldsymbol{x}_k^{(n)}|\boldsymbol{x}_{k-1}^{(n)}) v_{k-1}^{+(n)}(\boldsymbol{x}_{k-1}^{(n)}) \mathrm{d}\boldsymbol{x}_{k-1}^{(n)} + \gamma_k^{(n)}(\boldsymbol{x}_k^{(n)}) \Bigg] .$$

supporting the claimed result of Eq. (4.25). ■

C.10. PROOF OF EQ. (4.26)

Firstly, it can be shown that the PGFL of the posterior density produced by Bayes' rule is given as

$$G_{k|k}[h_1, \ldots, h_N | \boldsymbol{Z}_{1:k}] = \frac{\frac{\delta^m F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m}[0, h_1, \ldots, h_N]}{\frac{\delta^m F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m}[0, 1, \ldots, 1]} , \tag{C.1}$$

where $F[g, h_1, \ldots, h_N]$ is the joint PGFL given in Eq. (4.18) and $\boldsymbol{Z}_k = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m\}$ is the observation collected at epoch k. Proof of this claim can be found in Appendix C.11. Therefore, the desired PHD product, the PHD of the joint posterior is

$$\prod_{n=1}^{N} v_k^+(\boldsymbol{x}_k^{(n)}) = \frac{\delta G_{k|k}}{\delta \boldsymbol{x}_k^{(1)} \cdots \delta \boldsymbol{x}_k^{(N)}}[h_1, \ldots, h_N | \boldsymbol{Z}_{1:k}] \Big|_{h_1 = \cdots = h_N = 1} \tag{C.2}$$

$$= \frac{\frac{\delta^{m+N} F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m \delta \boldsymbol{x}_k^{(1)} \cdots \delta \boldsymbol{x}_k^{(N)}}[0, h_1, \ldots, h_N] \Big|_{h_1 = \cdots = h_N = 1}}{\frac{\delta^m F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m}[0, 1, \ldots, 1]} . \tag{C.3}$$

To compute the numerator of Eq. (C.1), recall the shorthand $F[g, h_1, \ldots, h_N] = F[g, h_{1:N}]$ and rearrange $F[g, h_{1:N}]$ to obtain

$$\begin{aligned} F[g, h_{1:N}] &\overset{\text{Eq. (4.18)}}{=} \int h^{\boldsymbol{X}_k^{(1:N)}} G[g|\boldsymbol{X}_k^{(1:N)}] \pi(\boldsymbol{X}_k^{(1:N)}|\boldsymbol{Z}_{1:k-1}) \delta \boldsymbol{X}_k^{(1:N)} \\ &\overset{\text{Eq. (4.22)}}{=} \int h^{\boldsymbol{X}_k^{(1:N)}} \left[\exp\{\lambda c[g] - \lambda\} \prod_{n=1}^{N} \left(q_{D,k}^{(n)} + p_{D,k}^{(n)} f_{D,k}^{(n)}[g] \right)^{\boldsymbol{X}_k^{(n)}} \right] \\ &\qquad \times \pi(\boldsymbol{X}_k^{(1:N)}|\boldsymbol{Z}_{1:k-1}) \delta \boldsymbol{X}_k^{(1:N)} . \end{aligned}$$

[1] This derivative follows precisely in the manner of [87]. Details are omitted here to avoid an overly lengthly illustration.

This can be rewritten as

$$F[g, h_{1:N}] = \exp\{\lambda c[g] - \lambda\} \prod_{n=1}^{N} \int h_n^{\boldsymbol{X}_k^{(n)}} \left(q_{D,k}^{(n)} + p_{D,k}^{(n)} f_{D,k}^{(n)}[g]\right)^{\boldsymbol{X}_k^{(n)}} \pi(\boldsymbol{X}_k^{(n)}|\boldsymbol{Z}_{1:k-1}) \delta\boldsymbol{X}_k^{(n)}$$
$$= \exp\{\lambda c[g] - \lambda\} \prod_{n=1}^{N} G_{k|k-1}^{(n)} \left[h_n \left(q_{D,k}^{(n)} + p_{D,k}^{(n)} f_{D,k}^{(n)}[g]\right)\right] ,$$

since, by definition,

$$G_{k|k-1}^{(n)}[h] = \int h^{\boldsymbol{X}_k^{(n)}} \pi(\boldsymbol{X}_k^{(n)}|\boldsymbol{Z}_{1:k-1}) \delta\boldsymbol{X}_k^{(n)} .$$

Write the set derivatives with respect to $\boldsymbol{x}_k^{(n)}$ in the direction of h_n as

$$\frac{\delta^N F}{\delta\boldsymbol{x}_k^{(1)} \cdots \delta\boldsymbol{x}_k^{(N)}}[g, h_{1:N}]$$
$$= \frac{\delta^N}{\delta\boldsymbol{x}_k^{(1)} \cdots \delta\boldsymbol{x}_k^{(N)}} \left\{ \exp\{\lambda c[g] - \lambda\} \prod_{n=1}^{N} G_{k|k-1}^{(n)} \left[h_n \left(q_{D,k}^{(n)} + p_{D,k}^{(n)} f_{D,k}^{(n)}[g]\right)\right] \right\} .$$

Since the sets are taken to be independent, each of the prior densities are (approximately) Poisson of the form

$$G_{k|k-1}^{(n)}[h_n|\boldsymbol{Z}_{1:k-1}] \approx \exp\left\{\mu^{(n)} \int h_n s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)} - \mu^{(n)}\right\} ,$$

where $\mu^{(n)}$ is the intensity[2] of the n^{th} PHD and $s^{(n)}(\boldsymbol{x}^{(n)}) = s^{(n)}(\boldsymbol{x}^{(n)}|\boldsymbol{Z}_{1:k-1})$ is that set's *a priori* spatial density, the product rule, and the general product rule yields

$$\frac{\delta^N F}{\delta\boldsymbol{x}_k^{(1)} \cdots \delta\boldsymbol{x}_k^{(N)}}[g, h_{1:N}]$$
$$= \frac{\delta^N}{\delta\boldsymbol{x}_k^{(1)} \cdots \delta\boldsymbol{x}_k^{(N)}} \left\{\exp\{\lambda c[g] - \lambda\}\right\} \prod_{n=1}^{N} G_{k|k-1}^{(n)} \left[h_n \left(q_{D,k}^{(n)} + p_{D,k}^{(n)} f_{D,k}^{(n)}[g]\right)\right]$$
$$+ \exp\{\lambda c[g] - \lambda\} \frac{\delta^N}{\delta\boldsymbol{x}_k^{(1)} \cdots \delta\boldsymbol{x}_k^{(N)}} \left\{ \prod_{n=1}^{N} G_{k|k-1}^{(n)} \left[h_n \left(q_{D,k}^{(n)} + p_{D,k}^{(n)} f_{D,k}[g]\right)\right] \right\}$$
$$= 0 + \exp\{\lambda c[g] - \lambda\} \frac{\delta^N}{\delta\boldsymbol{x}_k^{(1)} \cdots \delta\boldsymbol{x}_k^{(N)}} \left\{ \prod_{n=1}^{N} G_{k|k-1}^{(n)} \left[h_n \left(q_{D,k}^{(n)} + p_{D,k}^{(n)} f_{D,k}^{(n)}[g]\right)\right] \right\} .$$

[2] As in, $\mu^{(n)} = \int_{\mathcal{S}} v_k^{-(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)}$, where $\mathcal{S}$ is the entire state space of $\boldsymbol{x}^{(n)}$.

Here, the shorthand $G^{(n)}_{k|k-1}[h_n|\boldsymbol{Z}_{1:k-1}] \overset{\text{abbr.}}{=} G^{(n)}_{k|k-1}[h_n]$ has been adopted. Ultimately, using derivative properties of functionals, it turns out that

$$\frac{\delta^N F}{\delta \boldsymbol{x}_k^{(1)} \cdots \delta \boldsymbol{x}_k^{(N)}}[g, h_{1:N}] = \exp\{\lambda c[g] - \lambda\} \prod_{n=1}^{N} \Bigg\{ G^{(n)}_{k|k-1} \left[h_n \left(q^{(n)}_{D,k} + p^{(n)}_{D,k} f^{(n)}_{D,k}[g] \right) \right] \\ \times \left(q^{(n)}_{D,k} \mu^{(n)} s^{(n)} + p^{(n)}_{D,k} \mu^{(n)} s^{(n)} f^{(n)}_{D,k}[g] \right) \Bigg\}.$$

Given that all the derivatives in the direction of $h_1, \ldots, h_N$ have been taken, it is now permissible to set $h_1 = \cdots = h_N = 1$, yielding

$$\frac{\delta^N F}{\delta \boldsymbol{x}_k^{(1)} \cdots \delta \boldsymbol{x}_k^{(N)}}[g, 1, \ldots, 1] = \exp\{\lambda c[g] - \lambda\} \prod_{n=1}^{N} \Bigg\{ G^{(n)}_{k|k-1} \left[q^{(n)}_{D,k} + p^{(n)}_{D,k} f^{(n)}_{D,k}[g] \right] \\ \times \left(q^{(n)}_{D,k} \mu^{(n)} s^{(n)} + p^{(n)}_{D,k} \mu^{(n)} s^{(n)} f^{(n)}_{D,k}[g] \right) \Bigg\}.$$

What remains is to take m^{th} order set derivatives with respect to $\delta \boldsymbol{Z}_k = \delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m$ in the direction of g. Define the purely compressive notation

$$F^m = \frac{\delta^{m+N} F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m \delta \boldsymbol{x}_k^{(1)} \cdots \delta \boldsymbol{x}_k^{(N)}}[g, 1, \ldots, 1]$$

such that

$$F^m = \frac{\delta^m}{\delta \boldsymbol{z}_1 \cdots \boldsymbol{z}_m} \Bigg\{ \exp\{\lambda c[g] - \lambda\} \prod_{n=1}^{N} \Bigg\{ G^{(n)}_{k|k-1} \left[q^{(n)}_{D,k} + p^{(n)}_{D,k} f^{(n)}_{D,k}[g] \right] \\ \times \left(q^{(n)}_{D,k} \mu^{(n)} s^{(n)}(\boldsymbol{x}^{(n)}) + p^{(n)}_{D,k} \mu^{(n)} s^{(n)}(\boldsymbol{x}^{(n)}) f^{(n)}_{D,k}[g] \right) \Bigg\} \Bigg\}.$$

Taking the case that $m = 1$ and setting $g = 0$ yields

$$F^1 = \exp\{-\lambda\} \prod_{n=1}^{N} \left(\exp\left\{ -\mu^{(n)} \int p^{(n)}_{D,k} s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)} \right\} \mu^{(x)} s^{(n)}(\boldsymbol{x}^{(n)}) \right) \\ \times \Bigg\{ q^{\boldsymbol{N}}_{D,k} \left(\kappa_k(\boldsymbol{z}) + \sum_{n=1}^{N} \mu^{(n)} \int p^{(n)}_{D,k} g^{(n)}(\boldsymbol{z}_1|\boldsymbol{x}^{(n)}) s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)} \right) \\ + \sum_{n=1}^{N} q^{\boldsymbol{N}\backslash n}_{D,k} p^{(n)}_{D,k} g^{(n)}(\boldsymbol{z}_1|\boldsymbol{x}^{(n)}) \Bigg\},$$

where $\kappa_k(\boldsymbol{z}) = \lambda c(\boldsymbol{z})$ is the intensity of the clutter process, $q_{D,k}^{\boldsymbol{N}}$ denotes the product

$$q_{D,k}^{\boldsymbol{N}} = \prod_{n \in \boldsymbol{N}} q_{D,k}^{(n)}$$

with $\boldsymbol{N} = \{1, \ldots, N\}$, and since

$$\exp\left\{\mu^{(n)} \int q_{D,k}^{(n)} s^{(n)}(\boldsymbol{x}_k^{(n)}) \mathrm{d}\boldsymbol{x}_k^{(n)} - \mu^{(n)}\right\} = \exp\left\{-\mu^{(n)} \int p_{D,k}^{(n)} s^{(n)}(\boldsymbol{x}_k^{(n)}) \mathrm{d}\boldsymbol{x}_k^{(n)}\right\} .$$

Continuing in this fashion for $m = 2$, $m = 3$, etc. is a fairly grueling exercise, but ultimately one finds that for the general case that $|\boldsymbol{Z}_k| = m$, the numerator is given as

$$\begin{aligned} F^m = \exp\{-\lambda\} \left(\prod_{n=1}^{N} \exp\left\{-\mu^{(n)} \int p_{D,k}^{(n)} s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)}\right\} \mu^{(n)} s^{(n)}(\boldsymbol{x}^{(n)}) \right) \\ \times \left[q_{D,k}^{\boldsymbol{N}} a^{\boldsymbol{Z}_k} + \sum_{\boldsymbol{W} \subseteq_\emptyset \boldsymbol{N}} q_{D,k}^{\boldsymbol{N} \setminus \boldsymbol{W}} p_{D,k}^{\boldsymbol{W}} \sum_{\substack{\boldsymbol{Y} \subseteq_\emptyset \boldsymbol{Z}_k \\ |\boldsymbol{Y}| = |\boldsymbol{W}|}} a^{\boldsymbol{Z}_k \setminus \boldsymbol{Y}} \right. \\ \left. \times \sum_{1 \leq i_1 \leq \cdots \leq i_j \leq \ell} g^{(w_{i_1})}(\boldsymbol{y}_{i_1} | \boldsymbol{x}^{(w_{i_1})}) \cdots g^{(w_{i_j})}(\boldsymbol{y}_{i_j} | \boldsymbol{x}^{(w_{i_j})}) \right] , \end{aligned}$$

where a sum over $\boldsymbol{A} \subseteq_\emptyset \boldsymbol{B}$ denotes a sum over all nonempty subsets of $\boldsymbol{B}$ and

$$\begin{aligned} j &= |\boldsymbol{W}| \\ \ell &= |\boldsymbol{Y}| \\ a(\boldsymbol{z}) &= \kappa_k(\boldsymbol{z}) + \sum_{n=1}^{N} v^{-(n)} \left[p_{D,k}^{(n)} g^{(n)}(\boldsymbol{z} | \boldsymbol{x}^{(n)}) \right] \\ v^{-(n)}[h] &= \int h \cdot v_k^{-(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)} . \end{aligned}$$

The denominator of Eq. (C.1) is obtained by similarly taking derivatives with respect to $\boldsymbol{Z}_k$ in the direction of g. A reader able to produce the numerator should have little trouble demonstrating via proof by induction that the denominator is given as[3]

$$\begin{aligned}\frac{\delta^m F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m}[g, 1, \ldots, 1] \\ = \exp\{-\lambda\} \left(\prod_{n=1}^{N} \exp\left\{ \mu^{(n)} \int \left(q_{D,k}^{(n)} + p_{D,k}^{(n)} f_{D,k}^{(n)}[g] \right) s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)} - \mu^{(n)} \right\} \right) \\ \times \left(\kappa_k(\cdot) + \sum_{n=1}^{N} v^{-(n)} \left[p_{D,k}^{(n)} g^{(n)}(\cdot | \boldsymbol{x}^{(n)}) \right] \right)^{\boldsymbol{Z}_k}\end{aligned}$$

or, for $g = 0$ as desired,

$$\frac{\delta^m F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m}[0, 1, \ldots, 1] = \exp\{-\lambda\} \left(\prod_{n=1}^{N} \exp\left\{ -\mu^{(n)} \int p_{D,k}^{(n)} s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)} \right\} \right) a^{\boldsymbol{Z}_k} .$$

Substituting the expressions for the numerator and denominator into Eq. (C.2), recalling that $\mu^{(n)} s^{(n)}(\cdot) = v_k^{-(n)}(\cdot)$, noting that

$$\frac{a^{\boldsymbol{Z}_k \setminus \boldsymbol{Y}}}{a^{\boldsymbol{Z}_k}} = \frac{1}{a^{\boldsymbol{Y}}} ,$$

and utilizing the elementary symmetric function such that

$$\begin{aligned}\sum_{1 \le i_1 \le \cdots \le i_j \le \ell} \frac{g^{(w_{i_1})}(\boldsymbol{y}_{i_1} | \boldsymbol{x}^{(w_{i_1})}) \ldots g^{(w_{i_j})}(\boldsymbol{y}_{i_j} | \boldsymbol{x}^{(w_{i_j})})}{a(\boldsymbol{y}_{i_1}) \ldots a(\boldsymbol{y}_{i_j})}) \\ = \sigma_{\ell,j} \left(\frac{g^{(w_1)}(\boldsymbol{y}_1 | \boldsymbol{x}^{(w_1)})}{a(\boldsymbol{y}_1)}, \ldots, \frac{g^{(w_\ell)}(\boldsymbol{y}_\ell | \boldsymbol{x}^{(w_\ell)})}{a(\boldsymbol{y}_\ell)} \right)\end{aligned}$$

yields the posterior joint intensity and the claimed result. ■

[3] It should be noted that it may be possible to utilize Clark's general chain rule, as presented in [130], to avoid this proof by induction.

C.11. PROOF OF EQ. (4.23)/EQ. (C.1)

Bayes' rule states that the posterior joint multitarget density of all N RFS is given by

$$\pi(\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)}|\boldsymbol{Z}_{1:k}) = \frac{g(\boldsymbol{Z}_k|\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)})\pi(\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)}|\boldsymbol{Z}_{1:k-1})}{g_k(\boldsymbol{Z}_k|\boldsymbol{Z}_{1:k-1})}\,.$$

If $|\boldsymbol{Z}_k| = m$ at collection index k, take the m^{th} set derivative of the joint PGFL:

$$\begin{aligned}&\frac{\delta^m F}{\delta\boldsymbol{z}_1\cdots\delta\boldsymbol{z}_m}[g,h_1,\ldots,h_N]\\ &= \frac{\delta^m}{\delta\boldsymbol{z}_1\cdots\delta\boldsymbol{z}_m}\left\{\int\cdots\int h_1^{\boldsymbol{X}_k^{(1)}}\cdots h_N^{\boldsymbol{X}_k^{(N)}}\right.\\ &\qquad\left.\times G[g|\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)}]\pi(\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)}|\boldsymbol{Z}_{1:k-1})\delta\boldsymbol{X}_k^{(1)}\cdots\delta\boldsymbol{X}_k^{(N)}\right\}\end{aligned}$$

The Radon-Nikodym theorem permits the conclusion that setting $g = 0$ yields

$$\begin{aligned}\frac{\delta^m F}{\delta\boldsymbol{z}_1\cdots\delta\boldsymbol{z}_m}[0,h_1,\ldots,h_N] &= \int\cdots\int h_1^{\boldsymbol{X}_k^{(1)}}\cdots h_N^{\boldsymbol{X}_k^{(N)}}\\ &\quad\times g(\boldsymbol{Z}_k|\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)})\pi(\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)}|\boldsymbol{Z}_{1:k-1})\delta\boldsymbol{X}_k^{(1)}\cdots\delta\boldsymbol{X}_k^{(N)}\,,\end{aligned}$$

and, accordingly that subsequently setting $h_1 = \cdots = h_N = 1$ yields

$$\begin{aligned}&\frac{\delta^m F}{\delta\boldsymbol{z}_1\cdots\delta\boldsymbol{z}_m}[0,1,\ldots,1]\\ &\qquad= \int\cdots\int g(\boldsymbol{Z}_k|\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)})\pi(\boldsymbol{X}_k^{(1)},\ldots,\boldsymbol{X}_k^{(N)}|\boldsymbol{Z}_{1:k-1})\delta\boldsymbol{X}_k^{(1)}\cdots\delta\boldsymbol{X}_k^{(N)}\\ &\qquad= g(\boldsymbol{Z}_k|\boldsymbol{Z}_{1:k-1})\,,\end{aligned}$$

the denominator of Bayes' rule! Then, it can be seen that the ratio

$$\begin{aligned}
&\frac{\frac{\delta^m F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m}[0, h_1, \ldots, h_N]}{\frac{\delta^m F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m}[0, 1, \ldots, 1]} \\
&\quad = \frac{1}{g(\boldsymbol{Z}_k|\boldsymbol{Z}_{1:k-1})} \cdot \\
&\qquad \times \int \cdots \int h_1^{\boldsymbol{X}_k^{(1)}} \cdots h_N^{\boldsymbol{X}_k^{(N)}} \\
&\qquad \times g(\boldsymbol{Z}_k|\boldsymbol{X}_k^{(1)}, \ldots, \boldsymbol{X}_k^{(N)}) \pi(\boldsymbol{X}_k^{(1)}, \ldots, \boldsymbol{X}_k^{(N)}|\boldsymbol{Z}_{1:k-1}) \delta \boldsymbol{X}_k^{(1)} \cdots \delta \boldsymbol{X}_k^{(N)} \\
&\quad = \int \cdots \int h_1^{\boldsymbol{X}_k^{(1)}} \cdots h_N^{\boldsymbol{X}_k^{(N)}} \pi(\boldsymbol{X}_k^{(1)}, \ldots, \boldsymbol{X}_k^{(N)}|\boldsymbol{Z}_{1:k}) \delta \boldsymbol{X}_k^{(1)} \cdots \delta \boldsymbol{X}_k^{(N)} \\
&\quad \triangleq G[h_1, \ldots, h_N|\boldsymbol{Z}_{1:k}]
\end{aligned}$$

is the PGFL of the posterior distribution. That is, $G[h_1, \ldots, h_N|\boldsymbol{Z}_{1:k}]$ is the PGFL of Bayes' rule for N multitarget sets with potentially different state spaces. ■

C.12. PROOF OF EQ. (4.27)

To obtain an expression for the n^{th} marginal PHD, start with Eq. (4.26) and marginalize over the state spaces of all $\boldsymbol{x}_k^{(i)}$ such that $i \neq n$. That is, comptue

$$\begin{aligned}
&v_k^{+(n)}(\boldsymbol{x}_k^{(n)}) \int \cdots \int \prod_{i \neq n}^{N} v_k^{+(i)}(\boldsymbol{x}_k^{(i)}) \mathrm{d}\boldsymbol{x}_k^{(i)} \\
&= \int \cdots \int \left[q_{D,k}^{\boldsymbol{N}} + \sum_{\boldsymbol{W} \subseteq_\emptyset \boldsymbol{N}} q_{D,k}^{\boldsymbol{N} \setminus \boldsymbol{W}} p_{D,k}^{\boldsymbol{W}} \sum_{\substack{\boldsymbol{Y} \subseteq_\emptyset \boldsymbol{Z}_k \\ |\boldsymbol{Y}| = |\boldsymbol{W}|}} \sigma_{\ell,j} \left(\frac{g^{(w_1)}(\boldsymbol{y}_1|\boldsymbol{x}^{(w_1)})}{a(\boldsymbol{y}_1)}, \ldots, \frac{g^{(w_\ell)}(\boldsymbol{y}_\ell|\boldsymbol{x}^{(w_\ell)})}{a(\boldsymbol{y}_\ell)} \right) \right] \\
&\qquad \times v_k^{-(n)}(\boldsymbol{x}_k^{(n)}) \prod_{i \neq n}^{N} v_k^{-(i)}(\boldsymbol{x}_k^{(i)}) \mathrm{d}\boldsymbol{x}_k^{(i)} \, .
\end{aligned}$$

Since

$$\hat{N}_k^{+(i)} = \int v_k^{+(i)}(\boldsymbol{x}_k^{(i)}) \mathrm{d}\boldsymbol{x}_k^{(i)} \, ,$$

and expanding the elementary symmetric function, this can be rewritten as

$$v_k^{+(n)}(\boldsymbol{x}_k^{(n)})\prod_{i\neq n}^{N}\hat{N}_k^{+(i)} = \int\cdots\int\left[q_{D,k}^{\boldsymbol{N}} + \sum_{\boldsymbol{W}\subseteq\emptyset\boldsymbol{N}} q_{D,k}^{\boldsymbol{N}\setminus\boldsymbol{W}} p_{D,k}^{\boldsymbol{W}} \sum_{\substack{\boldsymbol{Y}\subseteq\emptyset\boldsymbol{Z}_k \\ |\boldsymbol{Y}|=|\boldsymbol{W}|}} \right.$$
$$\left. \times \sum_{1\leq i_1\leq\cdots\leq i_j\leq\ell} \frac{g^{(w_{i_1})}(\boldsymbol{y}_{i_1}|\boldsymbol{x}^{(w_{i_1})})\cdots g^{(w_{i_j})}(\boldsymbol{y}_{i_j}|\boldsymbol{x}^{(w_{i_j})})}{a(\boldsymbol{y}_{i_1})\cdots a(\boldsymbol{y}_{i_j})}\right] v_k^{-(n)}(\boldsymbol{x}_k^{(n)})\prod_{i\neq n}^{N} v_k^{-(i)}(\boldsymbol{x}_k^{(i)})\mathrm{d}\boldsymbol{x}_k^{(i)}\,.$$

Distributing and arranging terms produces

$$v_k^{+(n)}(\boldsymbol{x}_k^{(n)})\prod_{i\neq n}^{N}\hat{N}_k^{+(i)} = q_{D,k}(\boldsymbol{x}_k^{(n)})v_k^{-(n)}(\boldsymbol{x}_k^{(n)})\prod_{i\neq n}^{N} v_k^{-(i)}[q_{D,k}(\boldsymbol{x}_k^{(i)})]$$
$$+ \sum_{\boldsymbol{W}\subseteq\emptyset\boldsymbol{N}}\left(\prod_{i\in\boldsymbol{N}\setminus\boldsymbol{W}} v_k^{-(i)}[q_{D,k}(\boldsymbol{x}_k^{(i)})]\right)\int\cdots\int p_{D,k}^{\boldsymbol{W}}\sum_{\substack{\boldsymbol{Y}\subseteq\emptyset\boldsymbol{Z}_k \\ |\boldsymbol{Y}|=|\boldsymbol{W}|}}$$
$$\times \sum_{1\leq i_1\leq\cdots\leq i_j\leq\ell}\frac{g^{(w_{i_1})}(\boldsymbol{y}_{i_1}|\boldsymbol{x}^{(w_{i_1})})\cdots g^{(w_{i_j})}(\boldsymbol{y}_{i_j}|\boldsymbol{x}^{(w_{i_j})})}{a(\boldsymbol{y}_{i_1})\cdots a(\boldsymbol{y}_{i_j})} v_k^{-(n)}(\boldsymbol{x}_k^{(n)})\prod_{\substack{i\in\boldsymbol{W}\\ i\neq n}}^{N} v_k^{-(i)}(\boldsymbol{x}_k^{(i)})\mathrm{d}\boldsymbol{x}_k^{(i)}\,,$$

Note that in the sum over $1\leq i_1\leq\cdots\leq i_j\leq\ell$, there is exactly one term that corresponds to n and the remaining terms correspond to all sets $\boldsymbol{X}_k^{(i)}$ such that $i\neq n$.[4] Let this instance correspond to the index i_p, such that $w_{i_p}=n$. Therefore, one obtains

$$v_k^{+(n)}(\boldsymbol{x}_k^{(n)})\prod_{i\neq n}^{N}\hat{N}_k^{+(i)} = q_{D,k}(\boldsymbol{x}_k^{(n)})v_k^{-(n)}(\boldsymbol{x}_k^{(n)})\prod_{i\neq n}^{N} v_k^{-(i)}[q_{D,k}(\boldsymbol{x}_k^{(i)})]$$
$$+ \sum_{\boldsymbol{W}\subseteq\emptyset\boldsymbol{N}}\left(\prod_{i\in\boldsymbol{N}\setminus\boldsymbol{W}} v_k^{-(i)}[q_{D,k}(\boldsymbol{x}_k^{(i)})]\right)\sum_{\substack{\boldsymbol{Y}\subseteq\emptyset\boldsymbol{Z}_k \\ |\boldsymbol{Y}|=|\boldsymbol{W}|}}\sum_{1\leq i_1\leq\cdots\leq i_j\leq\ell} v_k^{-(n)}(\boldsymbol{x}_k^{(n)})$$
$$\times\frac{v_k^{-(w_{i_1})}[p_{D,k}^{(w_{i_1})}g^{(w_{i_1})}(\boldsymbol{y}_{i_1}|\boldsymbol{x}_k^{(w_{i_1})})]\cdots g^{(n)}(\boldsymbol{y}_{i_p}|\boldsymbol{x}^{(n)})\cdots v_k^{-(w_{i_j})}[p_{D,k}^{(w_{i_j})}g^{(w_{i_j})}(\boldsymbol{y}_{i_j}|\boldsymbol{x}_k^{(w_{i_j})})]}{a(\boldsymbol{y}_{i_1})\cdots a(\boldsymbol{y}_{i_p})\cdots a(\boldsymbol{y}_{i_j})}\,,$$

[4] This does, of course, assume that $n\in\boldsymbol{W}$, but the case that $n\in\boldsymbol{N}\setminus\boldsymbol{W}$ is trivially treated algorithmically and is discussed in the implementations discussion in Section 4.5.4.

or, using the elementary symmetric function

$$v_k^{+(n)}(\boldsymbol{x}_k^{(n)})\prod_{i\neq n}^{N}\hat{N}_k^{+(i)} = q_{D,k}(\boldsymbol{x}_k^{(n)})v_k^{-(n)}(\boldsymbol{x}_k^{(n)})\prod_{i\neq n}^{N}v_k^{-(i)}[q_{D,k}(\boldsymbol{x}_k^{(i)})]$$
$$+\sum_{\boldsymbol{W}\subseteq_\emptyset\boldsymbol{N}}\left(\prod_{i\in\boldsymbol{N}\backslash\boldsymbol{W}}v_k^{-(i)}[q_{D,k}(\boldsymbol{x}_k^{(i)})]\right)\sum_{\substack{\boldsymbol{Y}\subseteq_\emptyset\boldsymbol{Z}_k\\|\boldsymbol{Y}|=|\boldsymbol{W}|}}v_k^{-(n)}(\boldsymbol{x}_k^{(n)})$$
$$\times\,\sigma_{\ell,j}\left(\frac{v_k^{-(w_1)}[p_{D,k}^{(w_1)}g^{(w_1)}(\boldsymbol{y}_1|\boldsymbol{x}_k^{(w_1)})]}{a(\boldsymbol{y}_1)},\ldots,\frac{g^{(n)}(\boldsymbol{y}_{i_p}|\boldsymbol{x}^{(n)})}{a(\boldsymbol{y}_{i_p})},\ldots,\frac{v_k^{-(w_\ell)}[p_{D,k}^{(w_\ell)}g^{(w_\ell)}(\boldsymbol{y}_\ell|\boldsymbol{x}_k^{(w_\ell)})]}{a(\boldsymbol{y}_\ell)}\right),$$

Define the term

$$\phi(\boldsymbol{y}_i) = \frac{v_k^{-(w_i)}[p_{D,k}^{(w_i)}g^{(w_i)}(\boldsymbol{y}_i|\boldsymbol{x}_k^{(w_i)})]}{a(\boldsymbol{y}_i)}$$

such that this expression can be written as

$$v_k^{+(n)}(\boldsymbol{x}_k^{(n)}) =$$
$$\left(\prod_{i\neq n}^{N}\hat{N}_k^{+(i)}\right)^{-1}\left[q_{D,k}(\boldsymbol{x}_k^{(n)})\prod_{i\neq n}^{N}v_k^{-(i)}[q_{D,k}(\boldsymbol{x}_k^{(i)})]+\sum_{\boldsymbol{W}\subseteq_\emptyset\boldsymbol{N}}\left(\prod_{i\in\boldsymbol{N}\backslash\boldsymbol{W}}v_k^{-(i)}[q_{D,k}(\boldsymbol{x}_k^{(i)})]\right)\right.$$
$$\left.\times\sum_{\substack{\boldsymbol{Y}\subseteq_\emptyset\boldsymbol{Z}_k\\|\boldsymbol{Y}|=|\boldsymbol{W}|}}\sigma_{\ell,j}\left(\phi(\boldsymbol{y}_1),\ldots,\frac{g^{(n)}(\boldsymbol{y}_{i_p}|\boldsymbol{x}^{(n)})}{a(\boldsymbol{y}_{i_p})},\ldots,\phi(\boldsymbol{y}_\ell)\right)\right]v_k^{-(n)}(\boldsymbol{x}_k^{(n)}),$$

the claimed result. ■

C.13. PROOF OF EQ. (4.28)

Recall that the PGF (not PGFL) of Bayes' can be written as

$$G_{k|k}(h_1,\ldots,h_N|\boldsymbol{Z}_{1:k}) = \frac{\frac{\delta^m F}{\delta\boldsymbol{z}_1\cdots\delta\boldsymbol{z}_m}(0,h_1,\ldots,h_N)}{\frac{\delta^m F}{\delta\boldsymbol{z}_1\cdots\delta\boldsymbol{z}_m}(0,1,\ldots,1)},$$

where, in contrast to Eq. (C.1), $h_1, \ldots, h_N$ are now scalar (and not function) arguments. Therefore, the desired cardinality distribution is found via fundamental properties of PGFs using conventional derivatives as

$$\rho(i_1, \ldots, i_N) = \frac{1}{i_1! \cdots i_N!} \frac{\mathrm{d}^{i_1+\cdots+i_N} G_{k|k}}{\mathrm{d}h_1^{i_1} \cdots \mathrm{d}h_N^{i_N}}(h_1, \ldots, h_N | \boldsymbol{Z}_{1:k}) \Big|_{h_1=\cdots=h_N=0}.$$

Therefore, the derivatives

$$\frac{\delta^{m+i_1+\cdots+i_N} F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m \mathrm{d}h_1^{i_1} \cdots \mathrm{d}h_N^{i_N}}(0, h_1, \ldots, h_N)$$

are of principal interest. The first m set derivatives with respect to $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m$ in the direction of g are found as in Appendix C.10 to be

$$\frac{\delta^m F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m}(0, h_1, \ldots, h_N) = \exp\{-\lambda\} \prod_{n=1}^{N} \exp\left\{-h_n \mu^{(n)} \int p_{D,k}^{(n)} s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)}\right\} \times \prod_{j=1}^{m} \left(\kappa_k(\boldsymbol{z}_j) + \sum_{n=1}^{N} h_n \int p_{D,k}^{(n)} g^{(n)}(\boldsymbol{z}_j | \boldsymbol{x}^{(n)}) s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)}\right),$$

and, accordingly, the required denominator is handily obtained as

$$\frac{\delta^m F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m}(0, 1, \ldots, 1) = \exp\{-\lambda\} \prod_{n=1}^{N} \exp\left\{-\mu^{(n)} \int p_{D,k}^{(n)} s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)}\right\} \times \prod_{j=1}^{m} \left(\kappa_k(\boldsymbol{z}_j) + \sum_{n=1}^{N} \int p_{D,k}^{(n)} g^{(n)}(\boldsymbol{z}_j | \boldsymbol{x}^{(n)}) s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)}\right).$$

Now, the (conventional) derivatives with respect to h_n must be taken. To do so, first abbreviate

$$\frac{\delta^m F}{\delta \boldsymbol{z}_1 \cdots \delta \boldsymbol{z}_m}(0, h_1, \ldots, h_N) \overset{\text{abbr.}}{=} F^m$$

and recall that the general product rule is given as[5]

$$\left(\prod_{k=1}^{m} f_k\right)^{(n)} = \sum_{k_1+\cdots+k_m=n} \binom{n}{k_1,\ldots,k_m} \prod_{j=1}^{m} f_j^{(k_j)}$$

for some non-negative integers k_j. Then, define

$$a_{h_1,\ldots,h_N} = \prod_{j=1}^{m} \left(\kappa_k(\boldsymbol{z}_j) + \sum_{n=1}^{N} h_n \int p_{D,k}^{(n)} g^{(n)}(\boldsymbol{z}_j|\boldsymbol{x}^{(n)}) s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)} \right)$$

such that the derivative is given by

$$\begin{aligned}
&\frac{\mathrm{d}^{i_1+\cdots+i_N} F^m}{\mathrm{d}h_1^{i_1}\cdots\mathrm{d}h_N^{i_N}}(h_1,\ldots,h_N) \\
&= \exp\{-\lambda\} \prod_{n=1}^{N} \exp\left\{ -\mu^{(n)} \int p_{D,k}^{(n)} s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)} \right\} \\
&\times \left[\prod_{n=1}^{N} \frac{\left(\mu^{(n)} q_{D,k}^{(n)}\right)^{h_n}}{h_n!} a_{h_1,\ldots,h_N} + \sum_{n=1}^{N} \sum_{k_1+\cdots+k_m=i_n} \binom{h_n}{k_1,\ldots,k_m} \prod_{j=1}^{m} a_{h_1,\ldots,h_N}^{(k_j)} \right. \\
&+ \sum_{\substack{\boldsymbol{W}\subseteq\emptyset\boldsymbol{N} \\ |\boldsymbol{W}|=2}} \sum_{k_1^{(w_1)}+\cdots+k_m^{(w_1)}=i_{w_1}} \sum_{k_1^{(w_2)}+\cdots+k_m^{(w_2)}=i_{w_2}} \binom{h_{w_1}}{k_1^{(w_1)},\ldots,k_m^{(w_1)}} \binom{h_{w_2}}{k_1^{(w_2)},\ldots,k_m^{(w_2)}} \\
&\qquad\qquad \times \prod_{j=1}^{m} a_{h_1,\ldots,h_N}^{(k_j^{(w_1)})(k_j^{(w_2)})} \\
&+ \ldots \\
&\left. + \sum_{k_1^{(1)}+\cdots+k_m^{(1)}=i_1} \cdots \sum_{k_1^{(N)}+\cdots+k_m^{(N)}=i_N} \binom{h_1}{k_1^{(1)},\ldots,k_m^{(1)}} \cdots \binom{h_N}{k_1^{(N)},\ldots,k_m^{(N)}} \prod_{j=1}^{m} a_{h_1,\ldots,h_N}^{(k_j^{(1)})\cdots(k_j^{(N)})} \right],
\end{aligned}$$

where

$$a_{h_1,\ldots,h_N}^{(k_j^{(1)})\cdots(k_j^{(N)})} = \frac{\mathrm{d}^{k_j^{(N)}}}{\mathrm{d}h_N^{k_j^{(N)}}} \left\{ \frac{\mathrm{d}^{k_j^{(N-1)}}}{\mathrm{d}h_{N-1}^{k_j^{(N-1)}}} \left\{ \cdots \left\{ \frac{\mathrm{d}^{k_j^{(1)}} a_{h_1,\ldots,h_N}}{\mathrm{d}h_1^{k_j^{(1)}}} \right\} \right\} \right\}.$$

[5]For convenience, the n^{th} derivative of a function f is denoted here as $f^{(n)}$. There may be a risk of confusion between this and, say, $s^{(i)}(\boldsymbol{x}^{(i)})$, which is of course the spatial density of the i^{th} set. However, this convention reasonably compacts the notation and it will always be clear by inspection, since the derivatives are only concerned with the term $a_{h_1,\ldots,h_N}$.

Noting that

$$\begin{array}{ll} a_{h_1,\ldots,h_N}^{(k_j^{(1)})\cdots(k_j^{(N)})} = a_{h_1,\ldots,h_N} & \text{if } k_j^{(1)} = \cdots = k_j^{(N)} = 0 \\ a_{h_1,\ldots,h_N}^{(k_j^{(1)})\cdots(k_j^{(N)})} = \mu^{(n)} \int p_{D,k}^{(n)} g^{(n)}(\boldsymbol{z}_{k_j^{(n)}}|\boldsymbol{x}^{(n)}) s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)} & \text{if } k_j^{(n)} = 1 \text{ and } \sum_{n=1}^N k_j^{(n)} = 1 \\ a_{h_1,\ldots,h_N}^{(k_j^{(1)})\cdots(k_j^{(N)})} = 0 & \text{if } \sum_{n=1}^N k_j^{(n)} > 1 \end{array} ,$$

i.e. a great number of the terms within the combinatorics vanish and that $k_j = 0$ or 1 for all k_j, this can be rewritten as

$$\begin{aligned} &\frac{\mathrm{d}^{i_1+\cdots+i_N} F^m}{\mathrm{d}h_1^{i_1}\cdots\mathrm{d}h_N^{i_N}}(h_1,\ldots,h_N) \\ &= \exp\{-\lambda\} \prod_{n=1}^N \exp\left\{-\mu^{(n)} \int p_{D,k}^{(n)} s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)}\right\} \left[\prod_{n=1}^N \frac{\left(\mu^{(n)} q_{D,k}^{(n)}\right)^{h_n}}{h_n!} a_{h_1,\ldots,h_N} \right. \\ &\qquad \left. + \sum_{\boldsymbol{W}\subseteq_\emptyset \boldsymbol{N}} \sum_{\substack{\boldsymbol{Y}_1\uplus\cdots\uplus\boldsymbol{Y}_j\subseteq\boldsymbol{Z}_k \\ |\boldsymbol{Y}_1|=i_{w_1},\ldots,|\boldsymbol{Y}_j|=i_{w_j}}} \prod_{n=1}^j \left(\mu^{(w_n)} \int p_{D,k}^{(w_n)} g^{(w_n)}(\cdot|\boldsymbol{x}^{(w_n)}) s^{(w_n)}(\boldsymbol{x}^{(w_n)}) \mathrm{d}\boldsymbol{x}^{(w_n)}\right)^{\boldsymbol{Y}_n} \right] , \end{aligned}$$

Observing the shorthand

$$v_k^{-(n)}\left[p_{D,k}^{(w_n)} g^{(w_n)}(\cdot|\boldsymbol{x}_k^{(n)})\right] = \int p_{D,k}^{(w_n)} g^{(w_n)}(\cdot|\boldsymbol{x}^{(w_n)}) s^{(w_n)}(\boldsymbol{x}^{(w_n)}) \mathrm{d}\boldsymbol{x}^{(w_n)} ,$$

because $v_k^{-(n)} = \mu^{(n)} s^{(n)}(\boldsymbol{x}^{(n)})$, and setting $h_1 = \cdots = h_N = 0$ permits the numerator to be written as

$$\begin{aligned} &\frac{\mathrm{d}^{i_1+\cdots+i_N} F^m}{\mathrm{d}h_1^{i_1}\cdots\mathrm{d}h_N^{i_N}}(0,\ldots,0) \\ &\quad = \exp\{-\lambda\} \prod_{n=1}^N \exp\left\{-\mu^{(n)} \int p_{D,k}^{(n)} s^{(n)}(\boldsymbol{x}^{(n)}) \mathrm{d}\boldsymbol{x}^{(n)}\right\} \left[\kappa_k^{\boldsymbol{Z}_k} \prod_{n=1}^N \frac{\left(\mu^{(n)} q_{D,k}^{(n)}\right)^{h_n}}{h_n!} \right. \\ &\qquad \left. + \sum_{\boldsymbol{W}\subseteq_\emptyset \boldsymbol{N}} \sum_{\substack{\boldsymbol{Y}_1\uplus\cdots\uplus\boldsymbol{Y}_j\subseteq\boldsymbol{Z}_k \\ |\boldsymbol{Y}_1|=i_{w_1},\ldots,|\boldsymbol{Y}_j|=i_{w_j}}} \prod_{n=1}^j \left(v_k^{-(w_n)}\left[p_{D,k}^{(w_n)} g^{(w_n)}(\cdot|\boldsymbol{x}_k^{(w_n)})\right]\right)^{\boldsymbol{Y}_n} \right] , \end{aligned}$$

or, pulling out $\kappa_k^{\boldsymbol{Z}_k}$, as

$$\frac{\mathrm{d}^{i_1+\cdots+i_N}F^m}{\mathrm{d}h_1^{i_1}\cdots\mathrm{d}h_N^{i_N}}(0,\ldots,0)$$
$$= \exp\{-\lambda\}\prod_{n=1}^{N}\exp\left\{-\mu^{(n)}\int p_{D,k}^{(n)}s^{(n)}(\boldsymbol{x}^{(n)})\mathrm{d}\boldsymbol{x}^{(n)}\right\}\kappa_k^{\boldsymbol{Z}_k}\left[\prod_{n=1}^{N}\frac{\left(\mu^{(n)}q_{D,k}^{(n)}\right)^{h_n}}{h_n!}\right.$$
$$\left.+\sum_{\boldsymbol{W}\subseteq\emptyset\boldsymbol{N}}\sum_{\substack{\boldsymbol{Y}_1\uplus\cdots\uplus\boldsymbol{Y}_j\subseteq\boldsymbol{Z}_k\\|\boldsymbol{Y}_1|=i_{w_1},\ldots,|\boldsymbol{Y}_j|=i_{w_j}}}\prod_{n=1}^{j}\left(\frac{v_k^{-(w_n)}\left[p_{D,k}^{(w_n)}g^{(w_n)}(\cdot|\boldsymbol{x}_k^{(w_n)})\right]}{\kappa_k(\cdot)}\right)^{\boldsymbol{Y}_n}\right],$$

Substituting this and the previously found form of the denominator into the definition of $G_{k|k}(h_1,\ldots,h_N)$ with $h_1=\cdots=h_N=0$ and dividing by $i_1!\cdots i_N!$ yields the joint cardinality distribution and the claimed result. ■

C.14. PROOF OF EQ. (4.30)

Starting with the joint cardinality computation of Eq. (4.28), repeated here as,

$$\rho(i_1,\ldots,i_N)=\left[\prod_{n=1}^{N}\frac{\left(\mu^{(n)}q_{D,k}^{(n)}\right)^{i_n}}{i_n!}+\sum_{\boldsymbol{W}\subseteq\emptyset\boldsymbol{N}}\left(\prod_{n'\in\boldsymbol{N}\setminus\boldsymbol{W}}\frac{\left(\mu^{(n')}q_{D,k}^{(n')}\right)^{i_{n'}}}{i_{n'}!}\right)\right.$$
$$\left.\times\left(\sum_{\substack{\boldsymbol{Y}_1\uplus\cdots\uplus\boldsymbol{Y}_j\subseteq\boldsymbol{Z}_k\\|\boldsymbol{Y}_1|=i_{w_1},\ldots,|\boldsymbol{Y}_j|=i_{w_j}}}\prod_{n=1}^{j}\left(\frac{v_k^{-(w_n)}\left[p_{D,k}^{(w_n)}g^{(w_n)}(\cdot|\boldsymbol{x}_k^{(w_n)})\right]}{\kappa_k(\cdot)}\right)^{\boldsymbol{Y}_n}\right)\right]\left(\frac{\kappa_k(\cdot)}{a(\cdot)}\right)^{\boldsymbol{Z}_k},$$

one could obtain the n^{th} marginal cardinality distribution by summing over all realizable values of $i_c=\{0,\ldots,\infty\}$ such that $c\neq n$. That is,

$$\rho^{(n)}(i_n)=\sum_{c\neq n}^{N}\sum_{i_c=0}^{\infty}\rho(i_1,\ldots,i_N)\,.$$

This is tricky at best and, rather, the previously discovered form of $G_{k|k}(h_1,\ldots,h_N)$ can be used. For such a joint PGF, the marginal of the n^{th} variable is obtained by setting all other variables to 1; that is, the marginal cardinality distribution can be obtained with

$$\rho^{(n)}(i_n) = \frac{1}{i_n}\frac{\mathrm{d}^{i_n} G_{k|k}}{\mathrm{d}h_n^{i_n}}(1,\ldots,1,h_n,1,\ldots,1)\bigg|_{h_n=0}.$$

Setting $h_1 = \cdots = h_{n-1} = h_{n+1} = \cdots = h_N = 1$ and following the same procedure as in Appendix C.13 fairly handily produces the claimed result. Taking the required derivative is left as an exercise to the reader since it is, in fact, a much simpler case of the derivatives discussed in Appendix C.13. One must simply leverage the facts that

$$\sum_{k_1+\cdots+k_m=i_n}\prod_{j=1}^{m} a_{1,\ldots,1,0,1,\ldots,1}^{(k_j)} = \sum_{\substack{\boldsymbol{Y}\subseteq_\emptyset \boldsymbol{Z}_k \\ |\boldsymbol{Y}|=i_n}} \left(v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\cdot|\boldsymbol{x}_k^{(n)})\right]\right)^{\boldsymbol{Y}} \kappa_k(\boldsymbol{z}) + \sum_{\substack{i=1 \\ i\neq n}}^{N} v_k^{-(i)}\left[p_{D,k}^{(i)} g^{(i)}(\boldsymbol{z}|\boldsymbol{x}_k^{(i)})\right] = a(\boldsymbol{z})$$

to obtain the expression

$$\begin{aligned}\rho^{(n)}(i_n) = &\left(1 - \frac{v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\cdot|\boldsymbol{x}^{(n)})\right]}{a(\cdot)}\right)^{\boldsymbol{Z}_k} \\ &\times \left[\frac{\left(\mu^{(n)} q_{D,k}^{(n)}\right)^{i_n}}{i_n!} + \sum_{\substack{\boldsymbol{Y}\subseteq_\emptyset \boldsymbol{Z}_k \\ |\boldsymbol{Y}|=i_n}} \left(\frac{v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\cdot|\boldsymbol{x}^{(n)})\right]}{a(\cdot) - v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\cdot|\boldsymbol{x}^{(n)})\right]}\right)^{\boldsymbol{Y}}\right],\end{aligned}$$

and use of the elementary symmetric function such that

$$\begin{aligned}&\sum_{\substack{\boldsymbol{Y}\subseteq_\emptyset \boldsymbol{Z}_k \\ |\boldsymbol{Y}|=i_n}} \left(\frac{v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\cdot|\boldsymbol{x}^{(n)})\right]}{a(\cdot) - v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\cdot|\boldsymbol{x}^{(n)})\right]}\right)^{\boldsymbol{Y}} \\ &\quad = \sigma_{\ell,i_n}\left(\frac{v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\boldsymbol{y}_1|\boldsymbol{x}^{(n)})\right]}{a(\boldsymbol{y}_1) - v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\boldsymbol{y}_1|\boldsymbol{x}^{(n)})\right]},\ldots,\frac{v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\boldsymbol{y}_\ell|\boldsymbol{x}^{(n)})\right]}{a(\boldsymbol{y}_\ell) - v_k^{-(n)}\left[p_{D,k}^{(n)} g^{(n)}(\boldsymbol{y}_\ell|\boldsymbol{x}^{(n)})\right]}\right)\end{aligned}$$

yields the claimed result. ■

REFERENCES

[1] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960. doi: 10.1115/1.3662552.

[2] Rudolf E. Kalman and Richard S. Bucy. New results in linear filtering and prediction theory. *Journal of Basic Engineering*, 83(1):95–108, 1961.

[3] James S. McCabe and Kyle J. DeMars. Terrain-aided navigation with decentralized fusion and finite set statistics. *NAVIGATION, Journal of the Institute of Navigation*, 2018. (Submitted).

[4] James S. McCabe and Kyle J. DeMars. Multiple set filtering using probability hypothesis densities. In *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, 2018.

[5] James S. McCabe and Kyle J. DeMars. Fusion methodologies for orbit determination with distributed sensor networks. In *21st International Conference on Information Fusion (FUSION)*. IEEE, 2018.

[6] James S. McCabe and Kyle J. DeMars. Square-root consider filters with hyperbolic Householder reflections. *Journal of Guidance, Control, and Dynamics*, 41(10): 2098–2111, 2018. doi: 10.2514/1.G003417.

[7] James S. McCabe and Kyle J. DeMars. Robust, terrain-aided landing navigation through decentralized fusion and random finite sets. In *2018 AIAA Guidance, Navigation, and Control Conference*, 2018. doi: 10.2514/6.2018-1332.

[8] James S. McCabe and Kyle J. DeMars. Considering uncertain parameters in non-Gaussian estimation for single-target and multitarget tracking. *Journal of Guidance, Control, and Dynamics*, 40(9):2138–2150, 2017. doi: 10.2514/1.G002785.

[9] James S. McCabe and Kyle J. DeMars. Decentralized fusion with finite set statistics for landing navigation. In *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference*, 2017.

[10] James S. McCabe and Kyle J. DeMars. Feature-based robotic mapping with generalized labeled multi-Bernoulli filters for planetary landers. In *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference*, 2016. doi: 10.2514/6.2016-5565.

[11] James S. McCabe, Kyle J. DeMars, and Carolin Früh. Integrated detection and tracking for multiple space objects. In *Proceedings of the AAS/AIAA 24th Space Flight Mechanics Meeting*, Advances in the Astronautical Sciences, 2015.

[12] Thorvald N. Thiele. *Om anvendelse af mindste kvadraters methode i nogle tilfaelde, hvor en komplikation af visse slags uensartede tilfaeldige fejlkilder giver fejlene en "systematisk" karakter*. Det Kongelige Danske Videnskabernes Selskabs skrifter / Naturvidenskabelig og Mathematisk Afdeling. 1880.

[13] P. Swerling. *Proposed Stagewise Differential Correction Procedure for Satellite Tracking and Prediction*. RAND Corporation, 1958. Technical Report P-129.

[14] Richard H. Battin. Computational procedures for the navigational fix. Technical report, MIT Instrumentation Laboratory, April 1960. Appendix B of "Interplanetary Navigation System Study," Report R-273.

[15] Richard H. Battin. *An Introduction to the Mathematics and Methods of Astrodynamics*. AIAA Education Series. American Institute of Aeronautics & Astronautics, 1999.

[16] L.A. McGee. *Discovery of the Kalman filter as a practical tool for aerospace and industry*. National Aeronautics and Space Administration, 1985. doi: 10.1.1.467.52.

[17] Richard H. Battin. Oral history transcript. Technical report, 2000. Interviewed by Rebecca Wright.

[18] S. F. Schmidt. State space techniques applied to the design of a space navigation system. In *Joint Automatic Control Conference*, 1962.

[19] William M. Lear. The LM, powered flight, tracking-data processor. Technical report, 1968. NASA Internal Note 7224.5-8-R4.

[20] Alan Wylie. Personal interview conducted by James S. McCabe.

[21] William M. Lear. *On the Use of Ultrastable Oscillators and a Kalman Filter to Calibrate the Earth's Gravitational Field*. PhD thesis, Purdue University, 1965.

[22] William M. Lear. Multi-phase navigation program for the Space Shuttle Orbiter. Technical report, 1973. NASA Internal Note 73-FM-132.

[23] Renato Zanetti and Kyle J. DeMars. Joseph formulation of unscented and quadrature filters with application to consider states. *Journal of Guidance, Control, and Dynamics*, 36(6):1860–1864, 2013. doi: 10.2514/1.59935.

[24] William M. Lear. Speed and storage for conventional Kalman filters compared with square-root Kalman filters. Technical report, 1972. TRW Systems Group Memo 20029-6009-T0-01.

[25] R.S. Bucy and P.D. Joseph. *Filtering for Stochastic Processes with Applications to Guidance*. Interscience Tracts in Pure and Applied Mathematics. Interscience Publishers, 1968.

[26] Gene H Golub and John H Welsch. Calculation of Gauss quadrature rules. *Mathematics of computation*, 23(106):221–230, 1969.

[27] Ienkaran Arasaratnam and Simon Haykin. Cubature Kalman filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, 2009. doi: 10.1109/TAC.2009.2019800.

[28] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Proceedings of the SPIE - The International Society for Optical Engineering*, volume 3068, pages 182 – 193, April 1997. doi: 10.1117/12.280797.

[29] Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, March 2004. doi: 10.1109/JPROC.2003.823141.

[30] Renato Zanetti, Greg Holt, Robert Gay, Christopher DSouza, Jastesh Sud, Harvey Mamich, Michael Begley, Ellis King, and Fred D Clark. Absolute navigation performance of the Orion Exploration Flight Test 1. *Journal of Guidance, Control, and Dynamics*, 40(5):1106–1116, 2017. doi: 10.2514/1.G002371.

[31] Bernard A. Kriegsman and Yee-Chee Tao. Shuttle navigation system for entry and landing mission phases. *Journal of Spacecraft and Rockets*, 12(4):213–219, 1975. doi: 10.2514/3.56966.

[32] Athanasios Papoulis and S Unnikrishna Pillai. *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[33] Michael Athans, Richard Wishner, and Anthony Bertolini. Suboptimal state estimation for continuous-time nonlinear systems from discrete noisy measurements. *IEEE Transactions on Automatic Control*, 13(5):504–514, 1968. doi: 10.1109/TAC.1968.1098986.

[34] A.H. Jazwinski. *Stochastic Processes and Filtering Theory*. Mathematics in Science and Engineering. Elsevier Science, 1970. ISBN 9780080960906.

[35] Renato Zanetti, Kyle J DeMars, and Robert H Bishop. Underweighting nonlinear measurements. *Journal of guidance, control, and dynamics*, 33(5):1670–1675, 2010. doi: 10.2514/1.50596.

[36] John L Crassidis, Yang Cheng, Christopher K Nebelecky, and Adam M Fosbury. Decentralized attitude estimation using a quaternion covariance intersection approach. *The Journal of the Astronautical Sciences*, 57(1-2):113–128, 2009. doi: 10.1007/BF03321497.

[37] F. Landis Markley and John L. Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. Springer, 2014.

[38] Mohinder S Grewal and Angus P Andrews. *Kalman filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2014.

[39] Mohinder S Grewal and Angus P Andrews. Applications of Kalman filtering in aerospace 1960 to the present [historical perspectives]. *IEEE Control Systems*, 30 (3):69–78, 2010. doi: 10.1109/MCS.2010.936465.

[40] Richard H. Battin. *Astrounautical Guidance*. McGraw-Hill, New York, NY, USA, 1964. pp. 338-339.

[41] James E. Potter. New statistical formulas. Technical report, MIT Instrumentation Laboratory, April 1963. Space Guidance Analysis Memo #40.

[42] Stanley F. Schmidt. Computational techniques in Kalman filtering. Theory and Applications of Kalman Filtering, Harford House, London, Feb. 1970. NATO Advisory Group for Aerospace Research and Development.

[43] N. A. Carlson. Fast triangular formulation of the square root filter. *AIAA Journal*, 11(9):1259–1265, 1973.

[44] Alston S. Householder. Unitary triangularization of a nonsymmetric matrix. *Journal of the Association for Computing Machinery*, 5(4):339–342, October 1958. doi: 10.1145/320941.320947.

[45] P. Kaminski, A. Bryson, and S. Schmidt. Discrete square root filtering: A survey of current techniques. *IEEE Transactions on Automatic Control*, 16(6):727–736, Dec 1971. doi: 10.1109/TAC.1971.1099816.

[46] JF Bellantoni and KW Dodge. A square root formulation of the Kalman-Schmidt filter. *AIAA journal*, 5(7):1309–1314, 1967. doi: 10.2514/3.4189.

[47] G. J. Bierman. Measurement updating using the U-D factorization. In *IEEE Conference on Decision and Control including the 14th Symposium on Adaptive Processes*, pages 337–346, Dec 1975. doi: 10.1109/CDC.1975.270702.

[48] C.L. Thornton. *Triangular Covariance Factorizations for Kalman Filtering.* PhD thesis, 1976.

[49] Renato Zanetti, Greg Holt, Robert Gay, Christopher D'Souza, Jastesh Sud, Harvey Mamich, and Robert Gillis. Design and flight performance of the Orion prelaunch navigation system. *Journal of Guidance, Control, and Dynamics*, 40(9):2289–2300, 2017. doi: 10.2514/1.G002666.

[50] G. J. Bierman. *Factorization Methods for Discrete Sequential Estimation.* Academic Press, New York, NY, 1977. Chps. 7 and 8.

[51] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd Ed.).* Johns Hopkins University Press, Baltimore, MD, USA, 1996. Chps. 4 and 5.

[52] Rudolph van der Merwe and Eric A. Wan. The square-root unscented Kalman filter for state and parameter-estimation. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 3461–3464, 2001. doi: 10.1109/ICASSP.2001.940586.

[53] Rudolph van der Merwe. *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models.* PhD thesis, Oregon Health and Science University, Portland, Oregon, 2004.

[54] Jeffrey K Uhlmann. *Dynamic map building and localization: New theoretical foundations.* PhD thesis.

[55] Stanley F. Schmidt. Applications of state space methods to navigation problems. *in C. T. Leondes, Editor, Advanced Control Systems*, 3:293–340, 1966. doi: 10.1016/B978-1-4831-6716-9.50011-4.

[56] Michael Hough. Linear minimum variance filters for measurement bias characterization. *Journal of Guidance, Control, and Dynamics*, 36(1):337–342, 2012. doi: 10.2514/1.58968.

[57] Drew Woodbury and John Junkins. On the consider Kalman filter. AIAA Guidance, Navigation, and Control Conference, 2010. doi: 10.2514/6.2010-7752.

[58] Byron D. Tapley, Bob E. Schutz, and George H. Born. *Statistical Orbit Determination*. Elselvier Academic Press, New York, NY, 2004. Chapters 5 and 6.

[59] R. Fitzgerald. Divergence of the Kalman filter. *IEEE Transactions on Automatic Control*, 16(6):736–747, Dec 1971. doi: 10.1109/TAC.1971.1099836.

[60] Jeroen L. Geeraert and Jay W. McMahon. Square-root unscented Schmidt-Kalman filter. *Journal of Guidance, Control, and Dynamics*, 2017. doi: 10.2514/1.G002921.

[61] Jason Stauch and Moriba Jah. Unscented Schmidt-Kalman filter algorithm. *Journal of Guidance, Control, and Dynamics*, 38(1):117–123, 2017. doi: 10.2514/1.G000467.

[62] H. W. Sorenson and D. L. Alspach. Recursive Bayesian estimation using Gaussian sums. *Automatica*, 7(4):465–479, July 1971. doi: 10.1016/0005-1098(71)90097-5.

[63] C. Rader and A. Steinhardt. Hyperbolic Householder transformations. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(6):1589–1602, Dec 1986. doi: 10.1109/TASSP.1986.1164998.

[64] Angelika Bunse-Gerstner. An analysis of the HR algorithm for computing the eigenvalues of a matrix. *Linear Algebra and its Applications*, 35:155–173, 1981. doi: 10.1016/0024-3795(81)90271-8.

[65] A. Farina, B. Ristic, and D. Benvenuti. Tracking a ballistic target: comparison of several nonlinear filters. *IEEE Transactions on Aerospace and Electronic Systems*, 38 (3):854–867, Jul 2002. doi: 10.1109/TAES.2002.1039404.

[66] B. Ristic, A. Farina, D. Benvenuti, and M. S. Arulampalam. Performance bounds and comparison of nonlinear filters for tracking a ballistic object on re-entry. *IEE Proceedings - Radar, Sonar and Navigation*, 150(2):65–70, Apr 2003. ISSN 1350-2395. doi: 10.1049/ip-rsn:20030212.

[67] Y. C. Ho and RCKA Lee. A Bayesian approach to problems in stochastic estimation and control. *IEEE Transactions on Automatic Control*, 9(4):333–339, 1964. doi: 10.1109/TAC.1964.1105763.

[68] Daniel L. Alspach and Harold W. Sorenson. Nonlinear Bayesian estimation using Gaussian sum approximations. *IEEE Transactions on Automatic Control*, AC-17(4): 439–448, August 1972. doi: 10.1109/TAC.1972.1100034.

[69] B. N. Vo and Wing-Kin Ma. The Gaussian mixture probability hypothesis density filter. *IEEE Transactions on Signal Processing*, pages 4091–4104, 2006. doi: 10.1109/TSP. 2006.881190.

[70] B. T. Vo, B. N. Vo, and A. Cantoni. Analytic implementations of the cardinalized probability hypothesis density filter. *IEEE Transactions on Signal Processing*, 55(7): 3553–3567, 2007.

[71] B. T. Vo. *Random Finite Sets in Multi-Object Filtering*. PhD thesis, The University of Western Australia, 2008.

[72] B. T. Vo and B. N. Vo. Labeled random finite sets and multi-object conjugate priors. *IEEE Transactions on Signal Processing*, 61(13):3460–3475, July 2013. doi: 10.1109/TSP.2013.2259822.

[73] B. N. Vo, B. T. Vo, and D. Phung. Labeled random finite sets and the Bayes multi-target tracking filter. *IEEE Transactions on Signal Processing*, 62(24):6554–6567, Dec 2014. doi: 10.1109/TSP.2014.2364014.

[74] D. L. Alspach. A Gaussian sum approach to the multi-target identification-tracking problem. *Automatica*, 11(3):285–296, May 1975. ISSN 0005-1098. doi: 10.1016/0005-1098(75)90044-8.

[75] Kyle J. DeMars and Moriba K. Jah. Probabilistic initial orbit determination via Gaussian mixture models. *Journal of Guidance, Control, and Dynamics*, 36(5):1324–1335, September-October 2013. doi: 10.2514/1.59844.

[76] Joshua T. Horwood, Nathan D. Aragon, and Aubrey B. Poore. Gaussian sum filters for space surveillance: Theory and simulations. *Journal of Guidance, Control, and Dynamics*, 34(6):1839–1851, November–December 2011. doi: 10.2514/1.53793.

[77] Kyle J. DeMars, Robert H. Bishop, and Moriba K. Jah. Entropy-based approach for uncertainty propagation of nonlinear dynamical systems. *Journal of Guidance, Control, and Dynamics*, 2013. doi: 10.2514/1.58987.

[78] Robert W. Sittler. An optimal data association problem in surveillance theory. *Military Electronics, IEEE Transactions on*, 8(2):125–139, April 1964. doi: 10.1109/TME.1964.4323129.

[79] Y. Bar-Shalom. Tracking methods in a multitarget environment. *Automatic Control, IEEE Transactions on*, 23(4):618–626, Aug 1978. doi: 10.1109/TAC.1978.1101790.

[80] S.S. Blackman. Multiple hypothesis tracking for multiple target tracking. *Aerospace and Electronic Systems Magazine, IEEE*, 19(1):5–18, Jan 2004. doi: 10.1109/MAES.2004.1263228.

[81] Ronald P.S. Mahler. Nonadditive probability, finite-set statistics, and information fusion. In *Proceedings of the 34th IEEE Conference on Decision and Control*, volume 2, pages 1947–1952 vol.2, Dec 1995. doi: 10.1109/ICSAI.2012.6223121.

[82] I. R. Goodman, Ronald P. Mahler, and Hung T. Nguyen. *Mathematics of Data Fusion*. Kluwer Academic Publishers, Norwell, MA, USA, 1997. ISBN 0792346742.

[83] Ronald P.S. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House, Inc., Norwood, MA, USA, 2007.

[84] Kyle J. DeMars, Islam I. Hussein, Carolin Frueh, Moriba K. Jah, and R. Scott Erwin. Multiple object space surveillance tracking using finite set statistics. *Journal of Guidance, Control, and Dynamics*, 2015.

[85] M. C. Stein and C. L. Winter. An additive theory of probabilistic evidence accrual. Technical report, 1993. Los Alamos National Laboratories Report LA-UR-93-3336.

[86] M. C. Stein and R. R. Tenney. What's the difference between PHD and MHT? Technical report. Los Alamos National Laboratories Report, undated.

[87] Ronald P.S. Mahler. Multitarget Bayes filtering via first-order multitarget moments. *Aerospace and Electronic Systems, IEEE Transactions on*, 39(4):1152–1178, Oct 2003. doi: 10.1109/TAES.2003.1261119.

[88] Ronald P.S. Mahler. PHD filters of higher order in target number. *IEEE Transactions on Aerospace and Electronic Systems*, 43(4):1523–1543, October 2007. doi: 10.1109/TAES.2007.4441756.

[89] I. Schlangen, E. D. Delande, J. Houssineau, and D. E. Clark. A second-order PHD filter with mean and variance in target number. *IEEE Transactions on Signal Processing*, 66(1):48–63, Jan 2018. doi: 10.1109/TSP.2017.2757905.

[90] D. E. Clark and J. Bell. Bayesian multiple target tracking in forward scan sonar images using the PHD filter. *IEE Proceedings - Radar, Sonar and Navigation*, 152 (5):327–334, October 2005. ISSN 1350-2395. doi: 10.1049/ip-rsn:20045068.

[91] Branko Ristic, D Clark, Ba-Ngu Vo, and Ba-Tuong Vo. Adaptive target birth intensity for PHD and CPHD filters. *IEEE Transactions on Aerospace and Electronic Systems*, 48(2):1656–1668, 2012. doi: 10.1109/TAES.2012.6178085.

[92] M. Ulmke, O. Erdinc, and P. Willett. Gaussian mixture cardinalized PHD filter for ground moving target tracking. In *10th International Conference on Information Fusion*, pages 1–8, July 2007. doi: 10.1109/ICIF. 2007.4408105.

[93] G Battistelli, L Chisci, S Morrocchi, F Papi, A Benavoli, A Di Lallo, A Farina, and A Graziano. Traffic intensity estimation via PHD filtering. In *Proceedings of the 5th European Radar Conference, Amsterdam, The Netherlands*, pages 340–343, 2008.

[94] S. Reuter and K. Dietmayer. Pedestrian tracking using random finite sets. In *14th International Conference on Information Fusion*, pages 1–8, July 2011.

[95] Brandon A Jones. CPHD filter birth modeling using the probabilistic admissible region. *IEEE Transactions on Aerospace and Electronic Systems*, 54(3):1456–1469, 2018. doi: 10.1109/TAES.2018.2793378.

[96] K. A. LeGrand. Space-based relative multitarget tracking. Master's thesis, Missouri University of Science and Technology, Rolla, MO, 2015.

[97] Wenling Li, Yingmin Jia, Junping Du, and Fashan Yu. Gaussian mixture PHD filter for multi-sensor multi-target tracking with registration errors. *Signal Processing*, 93 (1):86–99, January 2013. doi: 10.1016/j.sigpro.2012.06.030.

[98] K. Punithakumar, T. Kirubarajan, and A Sinha. Multiple-model probability hypothesis density filter for tracking maneuvering targets. *IEEE Transactions on Aerospace and Electronic Systems*, 44(1):87–98, January 2008. doi: 10.1109/TAES.2008.4516991.

[99] Stephan Reuter. *Multi-object tracking using random finite sets*. PhD thesis, Universität Ulm, 2014.

[100] M. Beard, B. Tuong Vo, and B.-N. Vo. A Solution for Large-scale Multi-object Tracking. April 2018. arXiv:1804.06622 [stat.CO].

[101] Ronald P.S. Mahler. *Advances in Statistical Multisource-Multitarget Information Fusion*. Artech House, Inc., Norwood, MA, USA, 2014.

[102] John Mullane, Ba-Ngu Vo, Martin D Adams, and Ba-Tuong Vo. A random-finite-set approach to Bayesian SLAM. *IEEE Transactions on Robotics*, 27(2):268–282, 2011. doi: 10.1109/TRO.2010.2101370.

[103] Dominic Schuhmacher, Ba-Tuong Vo, and Ba-Ngu Vo. A consistent metric for performance evaluation of multi-object filters. *IEEE Transactions on Signal Processing*, 56:3447–3457, 2008.

[104] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[105] D. Vaman. TRN history, trends and the unused potential. In *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, Oct 2012. doi: 10.1109/DASC.2012.6382278.

[106] Andrew Johnson, Adnan Ansar, Larry Matthies, Nikolas Trawny, Anastasios Mourikis, and Stergios Roumeliotis. A general approach to terrain relative navigation for planetary landing. In *AIAA Infotech Aerospace 2007 Conference and Exhibit*. doi: 10.2514/6.2007-2854.

[107] Hordur Johannsson, Michael Kaess, Brendan Englot, Franz Hover, and John Leonard. Imaging sonar-aided navigation for autonomous underwater harbor surveillance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4396–4403. IEEE, 2010. doi: 10.1109/IROS.2010.5650831.

[108] German Ros, Sebastian Ramos, Manuel Granados, Amir Bakhtiary, David Vazquez, and Antonio M Lopez. Vision-based offline-online perception paradigm for autonomous driving. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 231–238. IEEE, 2015. doi: 10.1109/WACV.2015.38.

[109] Nikolas Trawny, Anastasios I Mourikis, Stergios I Roumeliotis, Andrew E Johnson, and James F Montgomery. Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks. *Journal of Field Robotics*, 24(5):357–378, 2007. doi: 10.1002/rob.20189.

[110] John Stephen Mullane, Ba-Ngu Vo, Martin David Adams, and Ba-Tuong Vo. *Random Finite Sets for Robot Mapping & SLAM: New Concepts in Autonomous Robotic Map Representations*. Springer Publishing Company, Incorporated, 2013. ISBN 3642268315, 9783642268311.

[111] K. Y. K. Leung, F. Inostroza, and M. Adams. Multifeature-based importance weighting for the PHD SLAM filter. *IEEE Transactions on Aerospace and Electronic Systems*, 52(6):2697–2714, December 2016. ISSN 0018-9251. doi: 10.1109/TAES.2016.150566.

[112] John L. Crassidis and John L. Junkins. *Optimal Estimation of Dynamic Systems*. CRC press, 2011.

[113] Tim Bailey, Simon Julier, and Gabriel Agamennoni. On conservative fusion of information with unknown non-gaussian dependence. In *15th International Conference on Information Fusion (FUSION)*, pages 1876–1883. IEEE, July 9–12 2012.

[114] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977. doi: 10.2307/2984875.

[115] Simon Julier and Jeffrey K. Uhlmann. *General Decentralized Data Fusion with Covariance Intersection*, chapter 14, pages 319–344. CRC Press, 2nd edition, 2009.

[116] Ronald P.S. Mahler. Detecting, tracking, and classifying group targets: a unified approach. In *Signal Processing, Sensor Fusion, and Target Recognition X*, volume 4380, pages 217–229. International Society for Optics and Photonics, 2001.

[117] Ronald P.S. Mahler. Extended first-order Bayes filter for force aggregation. In *Signal and Data Processing of Small Targets*, volume 4728, pages 196–208. International Society for Optics and Photonics, 2002.

[118] Anthony Jack Swain. *Group and extended target tracking with the probability hypothesis density filter*. PhD thesis, Heriot-Watt University, 2013.

[119] Léo Legrand, Audrey Giremus, Eric Grivel, Laurent Ratton, Bernard Joseph, and Clement Magnant. An hierarchical LMB/PHD filter for multiple groups of targets with coordinated motions. In *21st International Conference on Information Fusion (FUSION)*, Cambridge, United Kingdom, June 2018.

[120] Syed Ahmed Pasha, Ba-Ngu Vo, Hoang Duong Tuan, and Wing-Kin Ma. A gaussian mixture PHD filter for jump markov system models. *IEEE Transactions on Aerospace and Electronic systems*, 45(3), 2009. doi: 10.1109/TAES.2009.5259174.

[121] Yang Wei, Fu Yaowen, Long Jianqian, and Li Xiang. Joint detection, tracking, and classification of multiple targets in clutter using the PHD filter. *IEEE Transactions on Aerospace and Electronic Systems*, 48(4):3594–3609, 2012. doi: 10.1109/TAES.2012.6324744.

[122] Anthony Swain and Daniel Clark. Extended object filtering using spatial independent cluster processes. In *13th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2010.

[123] Umut Orguner, Christian Lundquist, and Karl Granström. Extended target tracking with a cardinalized probability hypothesis density filter. In *14th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2011.

[124] Karl Granstrom, Christian Lundquist, and Omut Orguner. Extended target tracking using a gaussian-mixture PHD filter. *IEEE Transactions on Aerospace and Electronic Systems*, 48(4):3268–3286, 2012.

[125] Karl Granstrom, Marcus Baum, and Stephan Reuter. Extended object tracking: Introduction, overview and applications. *arXiv preprint arXiv:1604.00970*, 2016.

[126] Ronald P.S. Mahler. On point processes in multitarget tracking. *arXiv preprint arXiv:1603.02373*, 2016.

[127] Roy L Streit. The probability generating functional for finite point processes, and its application to the comparison of PHD and intensity filters.

[128] Mahendra Mallick, Vikram Krishnamurthy, and Ba-Ngu Vo. *Integrated tracking, classification, and sensor management: theory and applications*. John Wiley & Sons, 2012.

[129] Roy Streit, Christoph Degen, and Wolfgang Koch. The pointillist family of multitarget tracking filters. *arXiv preprint arXiv:1505.08000*, 2015.

[130] Daniel Clark and Ronald Mahler. Generalized PHD filters via a general chain rule. In *15th International Conference on Information Fusion (FUSION)*, pages 157–164. IEEE, 2012.

[131] Andrew E Johnson and James F Montgomery. Overview of terrain relative navigation approaches for precise lunar landing. In *IEEE Aerospace Conference*, pages 1–10. IEEE, 2008. doi: 10.1109/AERO.2008.4526302.

[132] Dewey Adams, Thomas B Criss, and Uday J Shankar. Passive optical terrain relative navigation using APLNav. In *IEEE Aerospace Conference*, pages 1–9. IEEE, 2008. doi: 10.1109/AERO.2008.4526303.

[133] James Alexander, Yang Cheng, William Zheng, Nikolas Trawny, and Andrew Johnson. A terrain relative navigation sensor enabled by multi-core processing. In *IEEE Aerospace Conference*, pages 1–11. IEEE, 2012. doi: 10.1109/AERO.2012.6187003.

[134] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. Matthies. Vision-aided inertial navigation for spacecraft entry, descent, and landing. *IEEE Transactions on Robotics*, 25(2):264–280, April 2009. ISSN 1552-3098. doi: 10.1109/TRO.2009.2012342.

[135] Raman Maini and Himanshu Aggarwal. Study and comparison of various image edge detection techniques. *International journal of image processing (IJIP)*, 3(1):1–11, 2009. doi: 10.3965/j.issn.1934-6344.2011.02.083-090.

[136] Young-Keun Chang, Seok-Jin Kang, and Byung-Hoon Lee. High-accuracy image centroiding algorithm for cmos-based digital sun sensors. In *IEEE Sensors*, pages 329–336. IEEE, 2007. doi: 10.1109/ICSENS.2007.4388403.

[137] Renato Zanetti. *Advanced Navigation Algorithms for Precision Landing*. PhD thesis, The University of Texas at Austin, 2007.

[138] Paul G Savage. *Strapdown Analytics*. Strapdown, Maple Plain, MN. ISBN 9780971778603.

[139] Oliver J. Woodman. *An Introduction to Inertial Navigation*. Tech. Rep. UCAM-CL-TR-696, University of Cambridge, Computer Laboratory, August 2007.

[140] C. L. Parsons and E. J. Walsh. Off-nadir radar altimetry. *IEEE Transactions on Geoscience and Remote Sensing*, 27(2):215–224, Mar 1989. ISSN 0196-2892. doi: 10.1109/36.20300.

[141] Farzin Amzajerdian, Diego F. Pierrottet, Glenn D. Hines, Larry B. Petway, and Bruce W. Barnes. Fiber-based doppler lidar for vector velocity and altitude measurements. Optical Society of America, 2015. doi: 10.1364/LS.2015.LTu3I.2.

[142] Svenja Woicke, Andres Moreno Gonzalez, Isabelle El-Hajj, Jelle Mes, Martin Henkel, and Robert Klavers. Comparison of crater-detection algorithms for terrain-relative navigation. In *AIAA Guidance, Navigation, and Control Conference*, 2018. doi: 10.2514/6.2018-1601.

[143] Kyle J. DeMars. *Nonlinear Orbit Uncertainty Prediction and Rectification for Space Situational Awareness*. PhD thesis, The University of Texas at Austin, Austin, Texas, 2010.

VITA

James Samuel McCabe was born in St. Louis, MO and was raised by his parents, Michael and Stephanie Coleman, with his younger sister, Madisen Coleman. Upon his graduation from Fox High School in 2010, he pursued an aerospace engineering education at Missouri University of Science & Technology in Rolla, MO. In early 2013, he began conducting undergraduate research under Dr. Kyle J. DeMars, investigating novel strategies for initial orbit determination and multitarget tracking. Upon graduating *summa cum laude* with his Bachelor of Science degree in May 2014, he enrolled in the Ph.D. program at the same university under Dr. DeMars. In 2016, he received the NASA Space Technology Research Fellowship (NSTRF) to investigate novel navigation strategies, and he has participated in research efforts at NASA centers during several summers. He has submitted papers to more than ten conferences around the world, has been fortunate enough to be able to present at eight, and, in August 2017, received the John V. Breakwell Student Paper Award. In December 2018, James received his Ph.D. in Aerospace Engineering from Missouri University of Science & Technology.