JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 38, No. 2, February 2015

Gaussian Sum Reapproximation for Use in a Nonlinear Filter

Mark L. Psiaki*
Cornell University, Ithaca, New York 14853-7501
Jonathan R. Schoenberg†
Arzentech, Inc., Fishers, Indiana 46037
and
Isaac T. Miller‡
Coherent Navigation, Inc., San Mateo, California 94404

DOI: 10.2514/1.G000541

A new method has been developed to approximate one Gaussian sum by another. This algorithm is being developed as part of an effort to generalize the concept of a particle filter. In a traditional particle filter, the underlying probability density function is described by particles: Dirac delta functions with infinitesimal covariances. This paper develops an important component of a more general filter, which uses a Gaussian sum with "fattened" finite-covariance "blobs" (i.e., Gaussian components), which replace infinitesimal particles. The goal of such a filter is to save computational effort by using many fewer Gaussian components than particles. Most of the techniques necessary for this type of filter exist. The one missing technique is a resampling algorithm that bounds the covariance of each Gaussian component while accurately reproducing the original probability distribution. The covariance bounds keep the blobs from becoming too "fat" to ensure low truncation error in extended Kalman filter or unscented Kalman filter calculations. A new resampling algorithm is described, and its performance is studied using two test cases. The new algorithm enables Gaussian sum filter performance that is better than standard nonlinear filters when applied in simulation to a difficult seven-state estimation problem: the new filter's root mean square error is only 60% higher than the Cramer–Rao lower bound, whereas the next best filter's root mean square error is 370% higher.

I. Introduction

DIFFICULTIES can arise when solving certain nonlinear dynamic estimation problems. For example, a common solution algorithm known as the extended Kalman filter (EKF) has the potential to diverge or to yield suboptimal accuracy [1–4]. Various algorithms have been developed with the goal of improved convergence robustness or accuracy in the presence of strong nonlinearities, among them the unscented or sigma-points Kalman filter (UKF) [1,5], the particle filter (PF) [2], and the moving horizon estimator [6], also known as the backward-smoothing extended Kalman filter [3].

The PF is attractive for its simplicity and its theoretical guarantee of convergence to the optimal result in the limit of very many particles. The required number of particles to achieve a reasonable result, however, can become overwhelming for a state-space dimension as small as seven [4] or even as small as three or four [7].

A sensible generalization of the PF is to use Gaussian sums, also known as Gaussian mixtures, to represent probability density functions. In contrast, a PF effectively works with representations that are sums of Dirac delta functions. A Gaussian mixture generalizes this concept by using elements that have finite covariances instead of infinitesimal covariances. A sum of elements with nonnegligible width may be able to approximate a probability density function with many fewer terms than would be needed by a PF for the same degree of accuracy, as measured based on differences of multiple moments or based on the functional norm "distance" from the true probability density. Thus, a Gaussian mixture filter has the potential to solve the curse of dimensionality that causes a PF to become impractical for state-space dimensions above two or three.

Gaussian mixture filters have been studied extensively, and [8,9] are two early papers on this subject. The proposed filter, described in [10], is a modified version of typical Gaussian mixture filters that are contained in many references (e.g., [11–14]). It implements a separate standard EKF dynamic propagation and measurement update for each element of its Gaussian mixture. Psiaki [10] demonstrates that a good approximation of the full nonlinear Bayesian filter calculations can be implemented by using mixand-by-mixand EKF calculations along with static Gaussian multiple-model recalculation of the mixand weights after the measurement update, as in [15]. The present paper's resampling algorithm is used by the filter in [10] between its dynamic propagation and measurement update steps to restrict mixand covariances and limit the total number of mixands. A strength of the Gaussian mixture filter of [10] is the potential accuracy of its mixand-by-mixand EKF dynamic propagation and measurement update. If each element of the Gaussian mixture has a sufficiently small covariance, then the EKF calculations will be very accurate. This is true because a narrow distribution implies accuracy of the Taylor series approximations inherent in the EKF calculations. One of the first papers to highlight this important property of a Gaussian mixture filter that has mixands with small covariances is [9].

A number of previously published Gaussian mixture filtering schemes employ a resampling/reapproximation step. (Resampling and reapproximation are used interchangeably throughout this paper.) Some specifically seek to limit the covariances of the mixands [12–14]. Others use reapproximation for other reasons, such as limiting the number of mixands (e.g., [11]). Similar to the present paper, other efforts have concentrated solely on the problem of resampling in a way that bounds mixand covariance, but have not implemented full filters (e.g., [16–19]). All of the known previous efforts to reapproximate Gaussian mixtures subject to covariance restrictions employ one-dimensional approaches to reduce the covariance in a given mixand. They use an eigenvalue decomposition of a given mixand's covariance matrix and split the corresponding one-dimensional Gaussians in the resulting product into multiple one-dimensional Gaussians with smaller one-dimensional standard deviations. A number of the methods develop online tests for whether a mixand's covariance is too large for accurate local filter approximation and, therefore, in need of reapproximation by multiple mixands with

Presented as Paper 2010-7747 at the AIAA Guidance, Navigation, and Control Conference, Toronto, Canada, 2–5 August 2010; received 4 February 2014; revision received 2 May 2014; accepted for publication 28 July 2014; published online 2 January 2015.

*Professor, Sibley School of Mechanical and Aerospace Engineering. Associate Fellow AIAA.
†Senior Consulting Engineer.
‡Chief Technology Officer.

smaller covariance [12,13,17,19]. This one-dimensional splitting approach has drawbacks. Consider the required number of new mixands to approximate the original mixture closely while respecting a given covariance upper limit. This number might be very large due to the curse of dimensionality: The needed number of new mixands for a single original mixand could scale as M^n, where M is the number of one-dimensional mixands with reduced variance along each axis and n is the state-space dimension. There is no obvious way for most of these methods to exploit the possibility that severe nonlinearities occur only in a subspace of the state space. Strategies are employed in [18,19] that could result in splitting only in a subspace, thereby reducing the needed number of new mixands. The method of Bayramoglu et al. [18] enforces maximum covariances along each axis independently of the other axes. The method of Havlak and Campbell [19] uses a sigma-points fit criterion to choose a single axis of splitting, and it iterates these operations recursively in case multiple axes need splitting. Most of the developed Gaussian mixture reapproximation schemes include methods to reduce or bound the number of mixands (e.g., [12–14,18,19]). These methods perform merging as a separate step from the reapproximation that bounds mixand covariance, and the merging may not bound the merged covariances. Two of the methods employ the merging algorithm of [20]. None of these methods develop limiting conditions under which their reapproximated Gaussian mixtures converge to the original mixture.

In [21,22], weight adaptation schemes are developed that seek to improve Gaussian mixture dynamic propagation through continuous-time nonlinear differential equations. If the underlying mixands have covariances that are too large, then this approach can lead to unacceptably high model truncation errors when applying EKF or UKF calculations to each mixand, regardless of how the weights might be adapted during dynamic propagation. Horwood et al. [16] found that the weight adaptation approach used in [21,22] does not yield significant improvements for certain orbit determination problems.

Other Gaussian mixture reapproximation schemes appear in [20,23,24]. These algorithms' primary goal is to approximate an original mixture by a new one that has fewer elements. The ability to reduce the number of elements can arrest the growth of element numbers caused by forming products of state and noise mixtures, or it can lower the numbers as much as possible when computational resources are at a premium. Schoenberg et al. [25] found the algorithm of [20] to be effective for this purpose. The present algorithm retains reduction of the mixand count as a secondary goal, but its main goal is to develop a new approximate mixture whose elements all have covariances that satisfy a linear matrix inequality (LMI) upper bound. This latter goal is of primary importance when using Gaussian mixtures to generalize nonlinear particle filtering.

The present paper's contribution is a new Gaussian mixture reapproximation algorithm. It has three important properties: First, it chooses elements of the new mixture so that their covariances lie below an LMI upper bound, a bound that could vary with mixand mean or some other relevant quantity. This constraint is included as a means of ensuring that element-by-element EKF or UKF dynamic propagation and measurement update calculations will yield a sufficiently accurate approximation of the a posteriori probability density function when using this resampling algorithm within an approximate Bayesian nonlinear estimation algorithm. This LMI bounding approach obviates the need to implement multiple one-dimensional splittings of a given mixand along its covariance eigenvectors. Second, the new algorithm chooses new mixture elements and their weights in a way that seeks to approximate the original Gaussian mixture distribution accurately in the limit of a large number of new mixands. This paper demonstrates the accuracy of its reapproximation in the limiting case. Third, the new reapproximation algorithm tries to hold down the needed number of new mixands through a combination of strategies. These strategies include 1) maximization of new element covariances subject to the LMI constraint, 2) selection of new element means and weights in a way that tends to limit the number of new elements when some of the original elements already have sufficiently small covariances, and 3) fusion of elements if their Gaussian sum can be approximated well by a single Gaussian element while respecting the LMI covariance bound.

A significant property of the new reapproximation scheme is its asymptotic approach to the importance resampling procedure of a standard particle filter [2] in the limit of a very small upper bound on the covariances of the new elements. This asymptotic similarity causes the corresponding Gaussian mixture filter to be a natural generalization of the particle filter.

This paper develops and analyzes its new Gaussian mixture reapproximation algorithm in five main sections. Section II defines Gaussian mixtures using square-root information matrix notation and gives an overview of the Gaussian mixture reapproximation algorithm. Section III defines an LMI that bounds the covariances of the elements of the new Gaussian mixture. It develops an algorithm for choosing the covariance of a new element in a way that respects this limit, while deviating as little as possible from the covariance of a corresponding element of the original mixture. Section IV develops an algorithm for merging elements of the original mixture subject to a bound on the relative error between elements of the original mixture and their merged counterparts. The relative error is defined using the integral square difference (ISD) error metric between two Gaussian mixtures. Section V presents the algorithm that selects the means, covariances, and weights of the elements which constitute the mixture reapproximation. This section summarizes the entire reapproximation algorithm, and it demonstrates that the reapproximated distribution approaches the original one in the limit of a large number of new mixands. Section VI presents simulated example test results that illustrate the performance and usefulness of the new algorithm. Section VII contains conclusions.

II. Gaussian Mixture Probability Density Functions and Reapproximation Overview

A. Gaussian Mixture Definition

A Gaussian mixture is a weighted sum of Gaussian distributions. The ith element of the mixture, also called the ith mixand or the ith component, can by characterized by its square-root information matrix R_i and its mean $\boldsymbol{\mu}_i$. The element probability distribution is

$$\begin{aligned}\mathcal{N}_{\mathrm{sr}}(\boldsymbol{x};\boldsymbol{\mu}_i,R_i) &= \frac{|\det(R_i)|}{(2\pi)^{n/2}}\exp\{-0.5[R_i(\boldsymbol{x}-\boldsymbol{\mu}_i)]^T[R_i(\boldsymbol{x}-\boldsymbol{\mu}_i)]\}\\ &= \mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_i,R_i^{-1}R_i^{-T})\end{aligned} \tag{1}$$

where $\boldsymbol{x}$ and $\boldsymbol{\mu}_i$ are n-dimensional vectors and R_i is an n-by-n matrix. The covariance matrix of this distribution is $P_i = R_i^{-1}R_i^{-T}$, where the notation $(\,)^{-T}$ indicates the inverse of the transpose of the matrix in question. The notation $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},P)$ indicates the usual normal distribution in the vector $\boldsymbol{x}$ that has mean $\boldsymbol{\mu}$ and covariance matrix P. The notation $\mathcal{N}_{\mathrm{sr}}(\boldsymbol{x};\boldsymbol{\mu},R)$ indicates the same distribution in $\boldsymbol{x}$, except that its covariance is characterized by the square-root information matrix R in place of the covariance matrix P. This latter parameterization of the normal distribution will be used throughout the remainder of this paper. It allows a simple LMI solution in Sec. III, and it is consistent with the target application within a Gaussian mixture filter that uses numerically stable square-root information filter calculations.

Each element of a Gaussian mixture also has a weight w_i. Each weight must be nonnegative. The sum of all of the weights equals one. If there are N elements in the mixture, then

$$1 = \sum_{i=1}^{N} w_i \quad \text{and} \quad w_i \geq 0 \quad \text{for } i = 1, \ldots, N \tag{2}$$

Given the Gaussian component definition in Eq. (1) and weights that obey the constraints in Eq. (2), the corresponding Gaussian mixture is

$$p_{\mathrm{gm}}(\boldsymbol{x};w_1,\boldsymbol{\mu}_1,R_1,\ldots,w_N,\boldsymbol{\mu}_N,R_N) = \sum_{i=1}^{N} w_i\mathcal{N}_{\mathrm{sr}}(\boldsymbol{x};\boldsymbol{\mu}_i,R_i) \tag{3}$$

It is straightforward to show that this probability density function preserves the unit normalization constraint and that its mean and covariance are, respectively,

$$\boldsymbol{\mu}_{\text{gm}} = \sum_{i=1}^{N} w_i \boldsymbol{\mu}_i \quad \text{and}$$

$$P_{\text{gm}} = \sum_{i=1}^{N} w_i [R_i^{-1} R_i^{-T} + (\boldsymbol{\mu}_i - \boldsymbol{\mu}_{\text{gm}})(\boldsymbol{\mu}_i - \boldsymbol{\mu}_{\text{gm}})^T] \tag{4}$$

It is necessary to distinguish between two Gaussian mixture distributions in this paper. Suppose that one distribution, distribution "a," is characterized by the weights, mean values, and square-root information matrices $w_{ai}, \boldsymbol{\mu}_{ai}$, and R_{ai} for $i = 1, \ldots, N_a$. Similarly, suppose that another related distribution, distribution "b," is characterized by $w_{bj}, \boldsymbol{\mu}_{bj}$, and R_{bj} for $j = 1, \ldots, N_b$. The following shorthand notation is used to indicate these two distributions:

$$p_a(\boldsymbol{x}) = p_{\text{gm}}(\boldsymbol{x}; w_{a1}, \boldsymbol{\mu}_{a1}, R_{a1}, \ldots, w_{aN_a}, \boldsymbol{\mu}_{aN_a}, R_{aN_a}) = \sum_{i=1}^{N_a} w_{ai} \mathcal{N}_{\text{sr}}(\boldsymbol{x}; \boldsymbol{\mu}_{ai}, R_{ai}) \tag{5a}$$

$$p_b(\boldsymbol{x}) = p_{\text{gm}}(\boldsymbol{x}; w_{b1}, \boldsymbol{\mu}_{b1}, R_{b1}, \ldots, w_{bN_b}, \boldsymbol{\mu}_{bN_b}, R_{bN_b}) = \sum_{j=1}^{N_b} w_{bj} \mathcal{N}_{\text{sr}}(\boldsymbol{x}; \boldsymbol{\mu}_{bj}, R_{bj}) \tag{5b}$$

The goal of this paper is to develop a method that picks the parameters of distribution $b, N_b, w_{bj}, \boldsymbol{\mu}_{bj}$, and R_{bj} for $j = 1, \ldots, N_b$. It seeks to pick these parameters in a way that will cause $p_b(\boldsymbol{x})$ to be a good approximation of $p_a(\boldsymbol{x})$, while respecting an LMI lower bound on every $R_{bj}^T R_{bj}$ for $j = 1, \ldots, N_b$. The algorithm's LMI lower bound on $R_{bj}^T R_{bj}$ is an alternate means of enforcing an LMI upper bound on the covariance $P_{bj} = R_{bj}^{-1} R_{bj}^{-T}$.

Consider the example one-dimensional original Gaussian mixture $p_a(\boldsymbol{x})$ and its reapproximation $p_b(\boldsymbol{x})$ that are plotted along the horizontal axis of Fig. 1. The distribution $p_a(\boldsymbol{x})$ is the solid black curve, and its three weighted constituents are the three dotted gray curves. The resampled $p_b(\boldsymbol{x})$ is the dash-dotted gray curve. The standard deviations of the three $p_a(\boldsymbol{x})$ components are 0.42, 0.75, and 2.06, but each $p_b(\boldsymbol{x})$ component has a smaller standard deviation, 0.20. The goal of this paper is to develop an algorithm that generates $p_b(\boldsymbol{x})$ from $p_a(\boldsymbol{x})$ automatically in a way that makes its dash-dotted gray curve match the black curve of $p_a(\boldsymbol{x})$ accurately, while guaranteeing that each component of $p_b(\boldsymbol{x})$ has a sufficiently small covariance.

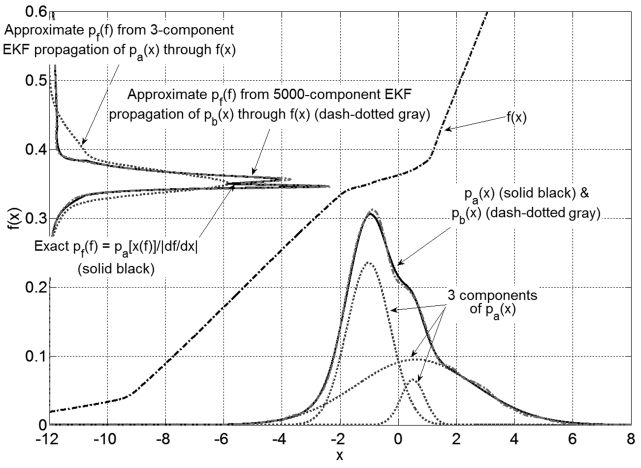

Fig. 1 Three-component original Gaussian mixture, a 5000-component reapproximation, and their propagation through a nonlinear function.

B. Overview of Gaussian Mixture Reapproximation Algorithm

This paper's reapproximation algorithm can be divided into three major phases. The first phase of the algorithm performs preprocessing calculations that produce candidate new square-root information matrices that respect LMI covariance bounds. It also calculates corresponding covariance matrix decrement square roots, which are used by the final resampling algorithm to compute the mean values of new mixands. The second phase applies a merging calculation to original Gaussian mixture $p_a(\boldsymbol{x})$ to create a modified mixture $p_{a'}(x)$. This phase attempts to reset some of the original weights to zero, whereas others are adjusted in a way that produces minimal distortion of the original distribution. The goal of this phase is to structure $p_{a'}(x)$ in a way that may reduce the number of mixands in the final resampled distribution. The last phase of the algorithm executes a procedure that determines the means, square-root information matrices, and weights of the mixands of the new Gaussian mixture $p_b(\boldsymbol{x})$.

The next three sections define the details of this resampling algorithm. Section III develops the LMI techniques needed to accomplish the first phase. Section IV develops the merging operations used by the second phase. Section V integrates these components into the complete reapproximation algorithm.

III. LMI Bounds on the Covariances of the New Mixture's Components

A. Covariance and Square-Root Information Matrix Bounds

This section defines and solves a linear matrix inequality. This solution is needed to compute the constrained covariances of the new mixture elements. The LMI is used to enforce the following lower bound on the information matrices of the elements of the new Gaussian mixture $p_b(\boldsymbol{x})$:

$$R_{bj}^T R_{bj} \geq R_{\min}^T R_{\min} \quad \text{for all } j = 1, \ldots, N_b \tag{6}$$

where the matrix inequality is defined in the sense that the symmetric matrix on the left minus the symmetric matrix on the right equals a positive semidefinite matrix.

This lower bound on the information matrix of each mixture element translates into an upper bound on each element's covariance: $P_{bj} = R_{bj}^{-1} R_{bj}^{-T} \leq R_{\min}^{-1} R_{\min}^{-T} = P_{\max}$. One can prove equivalence between this covariance inequality and Eq. (6) as follows: The latter matrix inequality is equivalent to $R_{\min} R_{bj}^{-1} R_{bj}^{-T} R_{\min}^T \leq I$. Equation (6) is equivalent to $R_{\min}^{-T} R_{bj}^T R_{bj} R_{\min}^{-1} \geq I$. The left-hand sides of these last two matrix inequalities are the inverses of each other. These last two inequalities are interchangeable because the first is true if and only if the symmetric matrix on its left-hand side has no eigenvalue greater than one, and the second is true if and only if its left-hand-side matrix has no eigenvalue less than one.

If the resampling algorithm must be constrained to choose the elements of $p_b(\boldsymbol{x})$ to have covariances less than $P_{\max}$, then it suffices to enforce the LMI in Eq. (6). This LMI provides a means of trying to ensure that element-by-element UKF or EKF operations on the mixture, as per [10], will yield a good approximation of optimal Bayesian nonlinear filtering because the corresponding local approximations of the filter's dynamics and measurement functions will be accurate over the likely range of state variability allowed by $P_{\max}$.

Choice of the bound $P_{\max}$ is problem dependent, and no general method has been developed for choosing this matrix based on the nonlinearities of the filtering problem's model functions. The choice of $P_{\max}$ should consider all of the nonlinearities in the filtering problem's dynamics and measurement models. It should be chosen so that a linear Taylor series approximation of each nonlinearity is reasonably accurate over the range of state perturbations Δx that respect the bound $\Delta \boldsymbol{x}^T P_{\max}^{-1} \Delta \boldsymbol{x} \leq \rho$ with the limit ρ chosen somewhere in the range of one to three.

Each R_{bj} square-root information matrix will be subject to one additional bound beyond that of Eq. (6). The Gaussian mixture resampling algorithm chooses the jth component of $p_b(\boldsymbol{x})$ with the

goal of improving the accuracy with which $p_b(\boldsymbol{x})$ approximates a particular element of $p_a(\boldsymbol{x})$, call it the ith element. For the resampling algorithm to work well, it is necessary that the covariance of the jth component of $p_b(\boldsymbol{x})$ not exceed the covariance of the corresponding ith component of $p_a(\boldsymbol{x})$. Otherwise, the resampling algorithm might not be able to produce a good approximation of the ith component of $p_a(\boldsymbol{x})$ because the new approximation's covariance will be no smaller than the smallest covariance of any of its components. Therefore, an appropriate additional bound on the new element's square-root information matrix is

$$R_{bj}^T R_{bj} \geq R_{ai}^T R_{ai} \tag{7}$$

This additional bound might appear to artificially restrict the width of $p_b(\boldsymbol{x})$'s approximation of the ith element of $p_a(\boldsymbol{x})$. This is not the case, however, because the full algorithm includes a compensatory widening of the resampled distribution through dispersion of the means of the new mixands that it uses to approximate the ith mixand of $p_a(\boldsymbol{x})$.

One might be tempted to impose an additional LMI upper bound on $R_{bj}^T R_{bj}$ to avoid unnecessary narrowness of the new mixands. Instead, an optimization of R_{bj} is used as a means of limiting the size of $R_{bj}^T R_{bj}$. It provides an effective "soft" upper limit and is easy to implement, as shown in the next subsection.

B. Optimal Solution to a Pair of Linear Matrix Inequalities

The algorithm for choosing R_{bj} seeks the matrix that satisfies the LMIs in Eqs. (6) and (7), while simultaneously minimizing two squared weighted-norm metrics: Trace$(R_{\min}^{-T} R_{bj}^T R_{bj} R_{\min}^{-1})$ and Trace $(R_{ai}^{-T} R_{bj}^T R_{bj} R_{ai}^{-1})$. This constrained minimization prevents R_{bj} from being any larger than needed, which yields the largest possible corresponding covariance matrix. This is a good way to choose R_{bj} because the largest possible covariance matrix tends to enable $p_b(\boldsymbol{x})$ to approximate $p_a(\boldsymbol{x})$ accurately, using the fewest possible mixands.

The optimal solution procedure for this LMI starts by computing the singular value decomposition of the matrix $R_{ai} R_{\min}^{-1}$:

$$U_{bj} S_{bj} V_{bj}^T = R_{ai} R_{\min}^{-1} \tag{8}$$

where U_{bj} and V_{bj} are orthonormal matrices and $S_{bj} = \text{diag}(\sigma_{bj1}, \ldots, \sigma_{bjn})$ is a diagonal matrix with the n positive singular values $\sigma_{bj1}, \ldots, \sigma_{bjn}$ on its diagonal.

If $\sigma_{bjk} \geq 1$, for all $k = 1, \ldots, n$, then the choice $R_{bj} = R_{ai}$ respects the LMIs in Eqs. (6) and (7) in an optimal manner. Otherwise, one forms the n-by-n diagonal matrix

$$\delta S_{bj\text{full}} = \begin{bmatrix} \sqrt{\max(1 - \sigma_{bj1}^2, 0)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{\max(1 - \sigma_{bjn}^2, 0)} \end{bmatrix} \tag{9}$$

Next, one deletes all of the zero-valued rows of $\delta S_{bj\text{full}}$ to form the matrix δS_{bj}. That is, row k of $\delta S_{bj\text{full}}$ is deleted for every k such that $\sigma_{bjk} \geq 1$. This latter matrix is then used to form the matrix

$$\delta R_{bj} = \delta S_{bj} V_{bj}^T R_{\min} \tag{10}$$

Finally, one uses orthonormal/upper-triangular (QR) factorization [26] to compute R_{bj} as follows:

$$Q_{bj} \begin{bmatrix} R_{bj} \\ 0 \end{bmatrix} = \begin{bmatrix} \delta R_{bj} \\ R_{ai} \end{bmatrix} \tag{11}$$

where Q_{bj} is an orthonormal matrix and R_{bj} is a square, upper-triangular matrix.

One can prove that this R_{bj} matrix satisfies the LMIs of Eqs. (6) and (7) if one recognizes the following implication of Eq. (11), which is that

$$R_{bj}^T R_{bj} = R_{ai}^T R_{ai} + \delta R_{bj}^T \delta R_{bj} \tag{12}$$

The LMI in Eq. (7) follows directly from this relationship. One can derive Eq. (6) by multiplying this relationship on the left by $R_{\min}^{-T}$ and on the right by $R_{\min}^{-1}$. One can then substitute in Eqs. (8) and (10) to show that $R_{\min}^{-T} R_{bj}^T R_{bj} R_{\min}^{-1} = V_{bj}(S_{bj}^T S_{bj} + \delta S_{bj}^T \delta S_{bj}) V_{bj}^T = V_{bj}(S_{bj}^T S_{bj} + \delta S_{bj\text{full}}^T \delta S_{bj\text{full}}) V_{bj}^T$. The last matrix expression within the parentheses is a diagonal matrix, all of whose diagonal elements are no less than one. Therefore, $R_{\min}^{-T} R_{bj}^T R_{bj} R_{\min}^{-1} \geq I$, which is equivalent to Eq. (6).

It is straightforward to show that the R_{bj} matrix of Eq. (11) minimizes both of the squared weighted-norm metrics Trace $(R_{\min}^{-T} R_{bj}^T R_{bj} R_{\min}^{-1})$ and Trace$(R_{ai}^{-T} R_{bj}^T R_{bj} R_{ai}^{-1})$ subject to the LMI bounds in Eqs. (6) and (7). Additionally, consider the eigenvalues of the two matrix differences $(R_{bj}^T R_{bj} - R_{\min}^T R_{\min})$ and $(R_{bj}^T R_{bj} - R_{ai}^T R_{ai})$. Both sets of eigenvalues are nonnegative, in accordance with the LMIs in Eqs. (6) and (7). Consider the union of the eigenvalues of these two positive semidefinite matrices, a set of $2n$ eigenvalues. It is straightforward to prove that n or more of these eigenvalues equal zero. These properties indicate that $R_{bj}^T R_{bj}$ is as close as possible, in some matrix sense, to $R_{\min}^T R_{\min}$ and to $R_{ai}^T R_{ai}$. Closeness to $R_{ai}^T R_{ai}$ tends to reduce the number of required new mixands for a given level of probability density approximation accuracy.

Note that the LMI solution R_{bj} is not unique. It can be left multiplied by any orthonormal matrix without changing any of the properties described in this subsection, except for upper triangularity. This nonuniqueness presents no problems. Any R_{bj} square-root information matrix with the given properties will serve for the development of the new Gaussian mixture $p_b(\boldsymbol{x})$.

C. Covariance Matrix Decrement Square Roots

A covariance matrix decrement must be computed for each original mixand of distribution $p_a(\boldsymbol{x})$. It is needed to define a modified distribution from which the mixand mean values of the new $p_b(\boldsymbol{x})$ distribution will be sampled in the overall reapproximation algorithm of Sec. V. This subsection defines the covariance matrix decrement and develops an algorithm to compute it.

The matrix δR_{bj} from Eq. (10) represents the square root of an increment to an information matrix. The corresponding covariance decrement is

$$dP_{aibj} = P_{ai} - P_{bj} = R_{ai}^{-1} R_{ai}^{-T} - R_{bj}^{-1} R_{bj}^{-T} = \delta Y_{aibj} \delta Y_{aibj}^T \tag{13}$$

This covariance decrement is positive semidefinite, and δY_{aibj} is its matrix square root. This square root is needed in Sec. V. It can be computed using following formula:

$$dY_{aibj} = R_{ai}^{-1} R_{ai}^{-T} \delta R_{bj}^T R_{djai}^{-1} \tag{14}$$

where the matrix R_{djai} is determined from the QR factorization

$$Q_{dj} \begin{bmatrix} R_{djai} \\ 0 \end{bmatrix} = \begin{bmatrix} R_{ai}^{-T} \delta R_{bj}^T \\ I \end{bmatrix} \tag{15}$$

with Q_{dj} being an orthonormal matrix and R_{djai} being a square, upper-triangular matrix.

One can prove that δY_{aibj} from Eq. (14) satisfies Eq. (13) by squaring the δY_{aibj} expression in Eq. (14), as on the right-hand side of Eq. (13), and by algebraically manipulating the result to show that it equals $R_{ai}^{-1} R_{ai}^{-T} - R_{bj}^{-1} R_{bj}^{-T}$. This manipulation requires the formula $R_{djai}^{-1} R_{djai}^{-T} = I - \delta R_{bj} R_{bj}^{-1} R_{bj}^{-T} \delta R_{bj}^T$. This latter formula can be proved by squaring both sides of Eq. (15) to show that $R_{djai}^T R_{djai} = I + \delta R_{bj} R_{ai}^{-1} R_{ai}^{-T} \delta R_{bj}^T$ and by using the matrix inversion lemma [27] along with a substitution based on Eq. (12). The next step of the proof of Eq. (13) replaces $R_{djai}^{-1} R_{djai}^{-T}$ in the formula for $\delta Y_{aibj} \delta Y_{aibj}^T$ with the equivalent expression given earlier. Finally, one performs several associative regroupings of matrix multiplications and two substitutions for $\delta R_{bj}^T \delta R_{bj}$ that are based on Eq. (12).

The covariance decrement square-root matrix δY_{aibj} has some interesting properties. It has only as many columns as δR_{bj} has rows. This number equals the number of singular values of S_{bj} that satisfy $\sigma_{bjk} < 1$. The matrix δY_{aibj} is not unique. It can be right multiplied by any orthonormal matrix without changing its satisfaction of Eq. (13). Any δY_{aibj} matrix that satisfies this equation will serve for Sec. V's resampling algorithm.

IV. Algorithm for Merging Elements of the Original Mixture

In the dynamic filtering application, it is possible that two or more elements of a Gaussian mixture will tend to converge to have nearly the same mean and covariance, as was discovered during the research that produced [28]. In such a situation, it is inefficient to maintain two or more separate mixands when a single mixand with an increased weight could accurately approximate the several original mixands. Therefore, a method has been developed to search for redundancies in the original Gaussian mixture $p_a(\boldsymbol{x})$ and to remove them to form a modified mixture called $p_{a'}(\boldsymbol{x})$.

This merging scheme constitutes an ad hoc method to try to restrict the number of mixands in the resampled distribution. It is unlikely to produce the optimal resampled distribution for a given bound on the number of mixands. It is implemented because it may substantially reduce the number of mixands in the final resampled distribution without significantly affecting the distribution's fidelity. This is especially likely when the algorithm is being used within a Bayesian Gaussian mixture filter which has converged to a narrow final distribution that yields very accurate state estimates [28].

A. Selection of Original Mixands for Possible Merging

Merging of Gaussian mixture $p_a(\boldsymbol{x})$ mixands is attempted only for those mixands with sufficiently small covariances [i.e., ones that already satisfy the information matrix lower-bound LMI in Eq. (6)]. This restriction on candidates for merging represents an ad hoc attempt to merge only those mixands that are likely to have converged on top of each other. It exploits the authors' experience that filter convergence tends to produce simultaneous overlapping of mixands and smallness of mixand covariances. As will be discussed in Sec. V, each original mixand with a sufficiently narrow covariance produces at most one resampled mixand. Therefore, the merging of such mixands in the original distribution can further reduce the number of mixands in the resampled distribution.

Suppose that the subset of mixands with the required properties for merging is used to define a new Gaussian mixture

$$\begin{aligned} p_c(\boldsymbol{x}) &= p_{gm}(\boldsymbol{x}; w_{c1}, \boldsymbol{\mu}_{c1}, R_{c1}, \ldots, w_{cN_c}, \boldsymbol{\mu}_{cN_c}, R_{cN_c}) \\ &= \sum_{k=1}^{N_c} w_{ck} \mathcal{N}_{\mathrm{sr}}(\boldsymbol{x}; \boldsymbol{\mu}_{ck}, R_{ck}) \end{aligned} \tag{16}$$

where the set of means and square-root information matrices used to define $p_c(x)$, $\{(\boldsymbol{\mu}_{c1}; R_{c1}), \ldots, (\boldsymbol{\mu}_{ck}; R_{ck}), \ldots, (\boldsymbol{\mu}_{cN_c}; R_{cN_c})\}$ is chosen to be the subset of $\{(\boldsymbol{\mu}_{a1}; R_{a1}), \ldots, (\boldsymbol{\mu}_{ai}; R_{ai}), \ldots, (\boldsymbol{\mu}_{aN_a}; R_{aN_a})\}$ whose elements obey

$$w_{ai} > 0 \quad \text{and} \quad R_{ai}^T R_{ai} \geq R_{\min}^T R_{\min} \tag{17}$$

Suppose that $i(k)$ defines the mapping from the $p_c(\boldsymbol{x})$ mixand index to the corresponding $p_a(\boldsymbol{x})$ mixand index so that $(\boldsymbol{\mu}_{ck}; R_{ck}) = (\boldsymbol{\mu}_{ai(k)}; R_{ai(k)})$ for $k = 1, \ldots, N_c$. Then the new weights used to define $p_c(\boldsymbol{x})$ are renormalized versions of the corresponding subset of the $p_a(\boldsymbol{x})$ weights:

$$w_{ck} = \frac{w_{ai(k)}}{(\sum_{l=1}^{N_c} w_{ai(l)})} \quad \text{for } k = 1, \ldots, N_c \tag{18}$$

Thus, $p_c(\boldsymbol{x})$ is a Gaussian mixture that represents a component of the original $p_a(\boldsymbol{x})$ distribution in the sense that $p_a(\boldsymbol{x})$ can be expressed as a weighted sum of $p_c(\boldsymbol{x})$ and another Gaussian mixture.

B. Merging Through Mixand Reweighting

The ad hoc merging algorithm attempts to find a new set of weights for $p_c(\boldsymbol{x})$ that includes some zero-valued weights and which results in the modified distribution

$$\begin{aligned} p_{c'}(\boldsymbol{x}) &= p_{gm}(\boldsymbol{x}; w_{c'1}, \boldsymbol{\mu}_{c1}, R_{c1}, \ldots, w_{c'N_c}, \boldsymbol{\mu}_{cN_c}, R_{cN_c}) \\ &= \sum_{k=1}^{N_c} w_{c'k} \mathcal{N}_{\mathrm{sr}}(\boldsymbol{x}; \boldsymbol{\mu}_{ck}, R_{ck}) \end{aligned} \tag{19}$$

The new weights are $w_{c'k}$ for $k = 1, \ldots, N_c$. The zero-valued weights correspond to mixands that will be dropped during the resampling process. In effect, the dropped mixands are merged into the remaining mixands.

The new weights must be chosen in a way that keeps the functional two norm of the difference between $p_c(\boldsymbol{x})$ and the new $p_{c'}(\boldsymbol{x})$ below a small relative error bound:

$$\begin{aligned} \|p_{c'}(\boldsymbol{x}) - p_c(\boldsymbol{x})\|_2 &= \left\{ \int_{-\infty}^{\infty} [p_{c'}(\boldsymbol{x}) - p_c(\boldsymbol{x})]^2 \, \mathrm{d}\boldsymbol{x} \right\}^{0.5} \\ &\leq \varepsilon \left\{ \int_{-\infty}^{\infty} [p_c(\boldsymbol{x})]^2 \, \mathrm{d}\boldsymbol{x} \right\}^{0.5} = \varepsilon \|p_c(\boldsymbol{x})\|_2 \end{aligned} \tag{20}$$

where ε is a small positive relative error limit, normally something on the order of 10^{-2}–10^{-4} or smaller. The integrals here and throughout the remainder of the paper are multidimensional integrals over the vector space associated with the corresponding differential hypervolume element, in this case the n-dimensional space associated with $\mathrm{d}\boldsymbol{x}$. This bound ensures that the dropping of mixands does not substantially alter the initial Gaussian mixture.

C. Definition and Use of Integral Square Difference

The probability density function relative error constraint in Eq. (20) can be reformulated by using the ISD between probability density functions. Given an original probability density function $p_a(\boldsymbol{x})$ and a candidate approximation $p_b(\boldsymbol{x})$, the ISD provides a measure of the accuracy with which $p_b(\boldsymbol{x})$ approximates $p_a(\boldsymbol{x})$ [23]. It is defined to be the integral of the square of the difference between these two probability density functions:

$$J_{\mathrm{ISD}} = \int_{-\infty}^{\infty} [p_a(\boldsymbol{x}) - p_b(\boldsymbol{x})]^2 \, \mathrm{d}\boldsymbol{x} \tag{21}$$

This quantity is nonnegative, and its square root is the functional two norm of the difference between the probability distributions:

$$\|p_a(\boldsymbol{x}) - p_b(\boldsymbol{x})\|_2 = \sqrt{J_{\mathrm{ISD}}} = \left\{ \int_{-\infty}^{\infty} [p_a(\boldsymbol{x}) - p_b(\boldsymbol{x})]^2 \, \mathrm{d}\boldsymbol{x} \right\}^{0.5} \tag{22}$$

A very small value of J_{ISD} indicates that $p_b(\boldsymbol{x})$ is a very good approximation of $p_a(\boldsymbol{x})$. Distribution $p_b(\boldsymbol{x})$ perfectly matches $p_a(\boldsymbol{x})$ if and only if $J_{\mathrm{ISD}} = 0$.

Williams and Maybeck [23] present analytic formulas for evaluating the integral in Eq. (21). The formulas used here are modified versions of those found in [23]. They account for the use of square-root information matrices in place of mixand covariance matrices. Suppose that one defines weight vectors for the two probability density functions $\boldsymbol{w}_a = [w_{a1}; w_{a2}; w_{a3}; \ldots; w_{aNa}]$ and $\boldsymbol{w}_b = [w_{b1}; w_{b2}; w_{b3}; \ldots; w_{bNb}]$. Then the integral in Eq. (21) can be written as a quadratic form in these two vectors:

$$J_{\mathrm{ISD}} = \boldsymbol{w}_a^T H_{aa} \boldsymbol{w}_a - 2\boldsymbol{w}_a^T H_{ab} \boldsymbol{w}_b + \boldsymbol{w}_b^T H_{bb} \boldsymbol{w}_b \tag{23}$$

where H_{aa}, H_{ab}, and H_{bb} are matrices with the respective dimensions N_a-by-N_a, N_a-by-N_b, and N_b-by-N_b. The matrices H_{aa} and H_{bb} are

symmetric and at least positive semidefinite. The elements of these matrices can be evaluated using the following formulas:

$$[H_{aa}]_{ik} = \int_{-\infty}^{\infty} \mathcal{N}_{\text{sr}}(\boldsymbol{x};\boldsymbol{\mu}_{ai}, R_{ai})\mathcal{N}_{\text{sr}}(\boldsymbol{x};\boldsymbol{\mu}_{ak}, R_{ak})\,\mathrm{d}\boldsymbol{x}$$
$$\text{for } i = 1, \ldots, N_a \quad \text{and} \quad k = 1, \ldots, N_a \tag{24a}$$

$$[H_{ab}]_{ij} = \int_{-\infty}^{\infty} \mathcal{N}_{\text{sr}}(\boldsymbol{x};\boldsymbol{\mu}_{ai}, R_{ai})\mathcal{N}_{\text{sr}}(\boldsymbol{x};\boldsymbol{\mu}_{bj}, R_{bj})\,\mathrm{d}\boldsymbol{x}$$
$$\text{for } i = 1, \ldots, N_a \quad \text{and} \quad j = 1, \ldots, N_b \tag{24b}$$

$$[H_{bb}]_{jl} = \int_{-\infty}^{\infty} \mathcal{N}_{\text{sr}}(\boldsymbol{x};\boldsymbol{\mu}_{bj}, R_{bj})\mathcal{N}_{\text{sr}}(\boldsymbol{x};\boldsymbol{\mu}_{bl}, R_{bl})\,\mathrm{d}\boldsymbol{x}$$
$$\text{for } j = 1, \ldots, N_b \quad \text{and} \quad l = 1, \ldots, N_b \tag{24c}$$

where the notation $[\,]_{ik}$ indicates the row-i/column-k element of the matrix in question.

The integrals in Eqs. (24a–24c) can be evaluated analytically by using the normalization property of a Gaussian distribution and the fact that the product of two Gaussian distributions is itself a Gaussian distribution, although not properly normalized [23]. These integrals take the general form

$$\int_{-\infty}^{\infty} \mathcal{N}_{\text{sr}}(\boldsymbol{x};\boldsymbol{\mu}_c, R_c)\mathcal{N}_{\text{sr}}(\boldsymbol{x};\boldsymbol{\mu}_d, R_d)\,\mathrm{d}\boldsymbol{x}$$
$$= \frac{|\det(R_c)||\det(R_d)|}{(2\pi)^{n/2}|\det(\bar{R}_{cd})|}\exp\{-0.5[\tilde{R}_{cd}(\boldsymbol{\mu}_c - \boldsymbol{\mu}_d)]^T[\tilde{R}_{cd}(\boldsymbol{\mu}_c - \boldsymbol{\mu}_d)]\} \tag{25}$$

where the n-by-n matrices $\bar{R}_{cd}$ and $\tilde{R}_{cd}$ are computed based upon the following QR factorization:

$$Q\begin{bmatrix}\bar{R}_{cd}\\ 0\end{bmatrix} = [Q_1, Q_2]\begin{bmatrix}\bar{R}_{cd}\\ 0\end{bmatrix} = \begin{bmatrix}R_c\\ R_d\end{bmatrix} \tag{26}$$

with Q being a $2n$-by-$2n$ orthonormal matrix and $\bar{R}_{cd}$ an n-by-n upper-triangular matrix. Q_1 equals the first n columns of Q, and Q_2 equals the last n columns. These matrices are used to compute

$$\tilde{R}_{cd} = Q_2^T\begin{bmatrix}R_c\\ 0\end{bmatrix} = -Q_2^T\begin{bmatrix}0\\ R_d\end{bmatrix} \tag{27}$$

Equations (25–27) have been derived using a lengthy, nonintuitive sequence of matrix/vector manipulations that have been omitted for the sake of brevity. In the special case where $R_c = R_d$, it suffices to use $\bar{R}_{cd} = \sqrt{2}R_c$ and $\tilde{R}_{cd} = (1/\sqrt{2})R_c$, and in this case, the Eq. (25) integral becomes

$$\int_{-\infty}^{\infty} \mathcal{N}_{\text{sr}}(\boldsymbol{x};\boldsymbol{\mu}_c, R_c)\mathcal{N}_{\text{sr}}(\boldsymbol{x};\boldsymbol{\mu}_d, R_c)\,\mathrm{d}\boldsymbol{x}$$
$$= \frac{|\det(R_c)|}{2^n\pi^{n/2}}\exp\{-0.25[R_c(\boldsymbol{\mu}_c - \boldsymbol{\mu}_d)]^T[R_c(\boldsymbol{\mu}_c - \boldsymbol{\mu}_d)]\} \tag{28}$$

Using the concept of the ISD, the probability density function relative error constraint in Eq. (20) becomes, after squaring both sides of the inequality and using the formulas in Eqs. (23–24c),

$$(\boldsymbol{w}_{c'} - \boldsymbol{w}_c)^T H_{cc}(\boldsymbol{w}_{c'} - \boldsymbol{w}_c) \le \varepsilon^2 \boldsymbol{w}_c^T H_{cc}\boldsymbol{w}_c \tag{29}$$

where $\boldsymbol{w}_c = [w_{c1}; w_{c2}; w_{c3}; \ldots; w_{cNc}]$, $\boldsymbol{w}_{c'} = [w_{c'1}; w_{c'2}; w_{c'3}; \ldots; w_{c'Nc}]$, and

$$[H_{cc}]_{ik} = \int_{-\infty}^{\infty} \mathcal{N}_{\text{sr}}(\boldsymbol{x};\boldsymbol{\mu}_{ci}, R_{ci})\mathcal{N}_{\text{sr}}(\boldsymbol{x};\boldsymbol{\mu}_{ck}, R_{ck})\,\mathrm{d}\boldsymbol{x}$$
$$\text{for } i = 1, \ldots, N_c \quad \text{and} \quad k = 1, \ldots, N_c \tag{30}$$

D. Determination of Merged Weights

Given these definitions, the merged weight vector $\boldsymbol{w}_{c'}$ is chosen by solving the following mixed real/integer optimization problem:

Find:

$$\boldsymbol{w}_{c'} = [w_{c'1}; w_{c'2}; w_{c'3}; \ldots; w_{c'N_c}] \tag{31a}$$

To maximize:

$$\text{number of zero-valued elements in } \boldsymbol{w}_{c'} \tag{31b}$$

Subject to:

$$1 = w_{c'1} + w_{c'2} + w_{c'3} + \ldots + w_{c'N_c} \tag{31c}$$

$$0 \le w_{c'k} \quad \text{for } k = 1, \ldots, N_c \tag{31d}$$

$$(\boldsymbol{w}_{c'} - \boldsymbol{w}_c)^T H_{cc}(\boldsymbol{w}_{c'} - \boldsymbol{w}_c) \le \varepsilon^2 \boldsymbol{w}_c^T H_{cc}\boldsymbol{w}_c \tag{31e}$$

The integer part of this constrained optimization problem comes from the fact that its performance index is the integer count of zero-valued elements of $\boldsymbol{w}_{c'}$.

The optimization problem in Eqs. (31a–31e) can be solved using brute-force techniques. The number of zero-valued elements in $\boldsymbol{w}_{c'}$ can never exceed $N_c - 1$ due to the weights' normalization constraint in Eq. (31c). A brute-force optimization works through all possible combinations of zero-valued elements of $\boldsymbol{w}_{c'}$, starting with the N_c combinations that have $N_c - 1$ zero-valued elements, next working with the $N_c(N_c - 1)/2$ combinations that have $N_c - 2$ zero-valued elements, etc. Thus, the N_c combinations that have the highest possible Eq. (31b) performance are tried first, followed by the $N_c(N_c - 1)/2$ combinations that have the next highest possible Eq. (31b) performance, etc. For each such combination, the values of the nonzero-valued $\boldsymbol{w}_{c'}$ elements are found that minimize the left-hand side of Eq. (31e). This minimization can be carried out using a few matrix-vector calculations because the minimized function is quadratic in the free elements of $\boldsymbol{w}_{c'}$, whereas the equality constraint in Eq. (31c) is linear in these elements [26]. If all such elements are positive and if the resulting minimum respects the inequality in Eq. (31e), then the optimum has been achieved, and there is no need to try any combinations with fewer zero-valued $\boldsymbol{w}_{c'}$ elements. All combinations with the same number of zero-valued $\boldsymbol{w}_{c'}$ elements are tried. If there are multiple possible feasible $\boldsymbol{w}_{c'}$ solutions that yield the same number of zero-valued elements, then the one with the lowest left-hand side of Eq. (31e) is selected because that one achieves the best fit to the original $p_c(\boldsymbol{x})$ distribution. If there are no feasible points with zero-valued elements of $\boldsymbol{w}_{c'}$, then the new weights are set equal to the old weights: $\boldsymbol{w}_{c'} = \boldsymbol{w}_c$.

It may be possible to solve the mixed integer/real optimization problem in Eqs. (31a–31e) without resorting to brute-force techniques, similar to the algorithm in section 4 of [29]. The computational cost of the brute-force method grows with increasing N_c in a combinatorial manner. Therefore, N_c should be restricted to a relatively small number if nothing better than the brute-force algorithm has been implemented.

E. Reweighting of Original Gaussian Mixture

The solution to the reweighting optimization problem in Eqs. (31a–31e) is used to define new weights for the original Gaussian mixture $p_a(\boldsymbol{x})$. These new weights are

$$w_{a'l} = \begin{cases} w_{al} & \text{if } l \notin \{i(1), i(2), i(3), \ldots, i(N_c)\} \\ \left(\sum_{j=1}^{N_c} w_{ai(j)}\right) w_{c'k} & \text{if there exists } k \in \{1, \ldots, N_c\} \text{ such that } l = i(k) \end{cases} \quad \text{for } l = 1, \ldots, N_a \tag{32}$$

The lower line in this formula ensures that

$$\sum_{k=1}^{N_c} w_{a'i(k)} = \sum_{k=1}^{N_c} w_{ai(k)} \tag{33}$$

because of the normalization constraint on the elements of $\boldsymbol{w}_{c'}$ in Eq. (31c). The effect of Eq. (33) is to retain the composite weight of the merged set of mixands relative to the other mixands. Given that the other mixand weights of $p_a(\boldsymbol{x})$ remain unchanged, Eq. (33) also ensures that the new weights in Eq. (32) are normalized.

The new weights in Eq. (32) are used to define a modified version of the original Gaussian mixture:

$$\begin{aligned} p_{a'}(\boldsymbol{x}) &= p_{gm}(\boldsymbol{x}; w_{a'1}, \boldsymbol{\mu}_{a1}, R_{a1}, \ldots, w_{a'N_a}, \boldsymbol{\mu}_{aN_a}, R_{aN_a}) \\ &= \sum_{i=1}^{N_a} w_{a'i} \mathcal{N}_{\text{sr}}(\boldsymbol{x}; \boldsymbol{\mu}_{ai}, R_{ai}) \end{aligned} \tag{34}$$

The constraint in Eq. (31e) ensures that this mixture will be very close to the original $p_a(\boldsymbol{x})$ mixture in the ISD sense. If the optimization problem in Eqs. (31a–31e) fails to find any mixand weights that can be set to zero safely, then $p_{a'}(\boldsymbol{x})$ will be identical to $p_a(\boldsymbol{x})$.

V. Sampling Algorithm that Generates the New Gaussian Mixture

The heart of the Gaussian mixture resampling algorithm involves choosing the elements of the new $p_b(\boldsymbol{x})$ and their weights so that this mixture will closely approximate the modified original mixture $p_{a'}(\boldsymbol{x})$. Recall that the elements and weights of $p_b(\boldsymbol{x})$ are, respectively, $\mathcal{N}_{\text{sr}}(\boldsymbol{x}; \boldsymbol{\mu}_{bj}, R_{bj})$ and w_{bj} for $j = 1, \ldots, N_b$, as per Eq. (5b). The design of $p_b(\boldsymbol{x})$ involves choosing the means, square-root information matrices, and weights $\boldsymbol{\mu}_{bj}$, R_{bj}, and w_{bj} for $j = 1, \ldots, N_b$. This section describes how these choices are made. It also summarizes the entire algorithm and demonstrates the convergence of $p_b(\boldsymbol{x})$ to $p_{a'}(\boldsymbol{x})$ in the limit of large N_b.

A. Preprocessing of Square-Root Information Matrices

The LMI solution calculations from Sec. III are carried out before the selection of new mixand components for distribution $p_b(\boldsymbol{x})$. Each new component of $p_b(\boldsymbol{x})$ is based on a component of $p_{a'}(\boldsymbol{x})$ through a type of sampling. Its mean and covariance will be based on the original mean and covariance of the corresponding $p_{a'}(\boldsymbol{x})$ element, but with modifications that flow from any square-root information matrix perturbation that is required to satisfy the LMIs in Eqs. (6) and (7). Therefore, it is necessary to precompute quantities associated with the perturbation of the square-root information matrix of each $p_{a'}(x)$ mixand.

For each $i = 1, \ldots, N_a$, the following LMI preprocessing calculations are performed: Use Eqs. (8–11) to compute R_{bj}, and rename this candidate distribution-b square-root information matrix $\tilde{R}_{ai}$. Next, use Eqs. (14) and (15) to compute δY_{aibj}, and rename this covariance decrement square-root matrix $\delta\tilde{Y}_{ai}$. Let n_{dYai} denote the number of columns of $\delta\tilde{Y}_{ai}$. It is no larger than n. If the original mixand square-root information matrix R_{ai} already satisfies the LMI in Eq. (6), then $\tilde{R}_{ai}$ will equal R_{ai}, $\delta\tilde{Y}_{ai}$ will be an empty matrix, and n_{dYai} will equal zero.

B. High-Level Steps of Gaussian Mixture Reapproximation Algorithm

The overall reapproximation algorithm has been outlined in Sec. II.B. It consists of three major phases that are defined by the following six steps:

1) Compute narrowed distribution square-root information matrices $\tilde{R}_{ai}$, covariance matrix decrement square roots $\delta\tilde{Y}_{ai}$, and the latter matrices' column dimensions n_{dYai} for $i = 1, \ldots, N_a$, as described in Sec. V.A. These calculations are carried out using Eqs. (8–11), (14), and (15), except that $\tilde{R}_{ai}$ takes the place of R_{bj} and $\delta\tilde{Y}_{ai}$ takes the place of δY_{aibj} in those equations; n_{dYai} equals the number of columns in $\delta\tilde{Y}_{ai}$, which will be zero if $\tilde{R}_{ai} = R_{ai}$.

2) Choose the set $\{(\boldsymbol{\mu}_{c1}; R_{c1}), \ldots, (\boldsymbol{\mu}_{ck}; R_{ck}), \ldots, (\boldsymbol{\mu}_{cN_c}; R_{cN_c})\}$ to be the subset of $\{(\boldsymbol{\mu}_{a1}; R_{a1}), \ldots, (\boldsymbol{\mu}_{ai}; R_{ai}), \ldots, (\boldsymbol{\mu}_{aN_a}; R_{aN_a})\}$ whose elements obey the conditions in Eq. (17). These define the truncated Gaussian mixture $p_c(\boldsymbol{x})$ in Eq. (16). If the initial count of elements in this set N_c is larger than some prespecified target $N_{c\text{target}}$, then discard from this set the $(N_c - N_{c\text{target}})$ elements that have the lowest corresponding weights in $p_a(\boldsymbol{x})$.

3) Solve the optimization problem in Eqs. (31a–31e) to compute the modified weights that define the modified truncated Gaussian mixture $p_{c'}(\boldsymbol{x})$.

4) Use Eq. (32) to compute the modified weights that define the merged modification of the original Gaussian mixture $p_{a'}(\boldsymbol{x})$.

5) Select $N_{b\text{target}}$, the target number of mixands in Gaussian mixture $p_b(\boldsymbol{x})$, initialize $l_0(i) = 0$ for $i = 1, \ldots, N_a$, and initialize empty arrays for $\boldsymbol{\mu}_{bj}$, R_{bj}, and w_{unbj} that will store, respectively, the means, square-root information matrices, and unnormalized weights that will be generated by the nine-step algorithm presented in Sec. V.C.

6) Execute the nine-step algorithm that is defined in Sec. V.C.

The first major algorithm phase is step 1, the LMI precomputation phase. Steps 2–4 implement phase two, the mixand merging phase. Steps 5 and 6 correspond to phase three, the final determination of new mixands. At the end of step 6, the new mixture $p_b(\boldsymbol{x})$ is completely defined via $\boldsymbol{\mu}_{bj}$, R_{bj}, and w_{bj} for $j = 1, \ldots, N_b$.

C. Sampling-Based Approach for Choosing Means, Square-Root Information Matrices, and Weights of the Resampled Gaussian Mixture

The algorithm for choosing the components and weights of $p_b(\boldsymbol{x})$, as defined in Eq. (5b), amounts to a sampling algorithm from a Gaussian mixture that is a modified version of $p_{a'}(\boldsymbol{x})$. This modified version has the same means and weights, $\boldsymbol{\mu}_{ai}$ and $w_{a'i}$ for $i = 1, \ldots, N_a$, but it has modified covariances. In place of the covariance $P_{ai} = R_{ai}^{-1} R_{ai}^{-T}$, each mixand's covariance is the narrowed value $\delta\tilde{P}_{ai} = \delta\tilde{Y}_{ai}\delta\tilde{Y}_{ai}^T$. If $\delta\tilde{Y}_{ai}$ is an empty matrix and $n_{dYai} = 0$, then the corresponding covariance is $\delta\tilde{P}_{ai} = 0$; that is, the modified mixand is a Dirac delta function.

The resampling algorithm starts with a target number of $p_b(\boldsymbol{x})$ mixands $N_{b\text{target}}$, and it initializes and updates various quantities during its sampling procedure. These quantities include the current total number of actual new mixands j, along with the mean, square-root information matrix, and unnormalized weight of each of these new mixands. These latter quantities are, respectively, $\boldsymbol{\mu}_{bl}$, R_{bl}, and w_{unbl} for $l = 1, \ldots, j$. Another set of stored quantities are the indices of the first new mixands of distribution $p_b(\boldsymbol{x})$ that have been sampled from given mixands of distribution $p_{a'}(\boldsymbol{x})$. Let these indices be designated as $l_0(i)$ for $i = 1, \ldots, N_a$. These indices are initialized to the values $l_0(i) = 0$ for $i = 1, \ldots, N_a$ to indicate that no mixands of distribution $p_b(\boldsymbol{x})$ have yet been sampled from the corresponding mixands of distribution $p_{a'}(\boldsymbol{x})$.

Given these definitions, the resampling algorithm executes the following steps.

1) Set $j = 1$ and seed random number generators.

2) Use a random number generator to draw a scalar sample β from the uniform distribution $\mathcal{U}[0, 1]$ and find the unique value of i in the range $1 - N_a$, such that

$$\left\{ \begin{array}{ll} 0 & \text{if } i = 1 \\ \sum_{k=1}^{i-1} w_{a'k} & \text{if } i > 1 \end{array} \right\} \leq \beta < \left\{ \begin{array}{ll} \sum_{k=1}^{i} w_{a'k} & \text{if } i < N_a \\ 1 + \varepsilon & \text{if } i = N_a \end{array} \right\} \quad (35)$$

where ε is any small positive number, not necessarily the same number as is used in Eq. (31e).

3) If $n_{dYai} > 0$, then use a random number generator to sample $\boldsymbol{\eta}$ from the n_{dYai}-dimensional, zero-mean, identity-covariance Gaussian distribution $\mathcal{N}(\boldsymbol{\eta}; 0, I)$, set $\boldsymbol{\mu}_{bj} = \boldsymbol{\mu}_{ai} + \delta\tilde{Y}_{ai}\boldsymbol{\eta}$, and skip to step 6. Otherwise, continue to step 4.

4) If $l_0(i) = 0$, then set $\boldsymbol{\mu}_{bj} = \boldsymbol{\mu}_{ai}$ and skip to step 6. Otherwise, continue to step 5.

5) Set $m = l_0(i)$, increment w_{unbm} by one, decrement j by one, and skip to step 8.

6) Set $R_{bj} = \tilde{R}_{ai}$ and $w_{unbj} = 1$.

7) If $l_0(i) = 0$, then reassign $l_0(i) = j$.

8) If $\sum_{l=1}^{j} w_{unbl} < N_{b\text{target}}$ then increment j by one and return to step 2. Otherwise, continue to step 9.

9) Set $N_b = j$, set $w_{bk} = w_{unbk}/N_{b\text{target}}$ for $k = 1, \ldots, N_b$, and terminate.

The steps of this algorithm can be interpreted as follows: Step 2 samples the discrete probability "mass" function over the Gaussian mixture weights to select mixand i from distribution $p_{a'}(\boldsymbol{x})$. Mixand i normally will be used to define the new jth mixand of distribution $p_b(\boldsymbol{x})$. If $n_{dYai} = 0$ and $l_0(i) > 0$, however, then the selection of mixand i will result only in an increase of the weight for already existing mixand $l_0(i)$.

Steps 3 or 4 are executed if there is a new mixand element, followed by steps 6 and 7. Step 3 defines the mean of the new jth mixand by sampling from a narrowed Gaussian distribution centered at the mean of the corresponding $p_{a'}(\boldsymbol{x})$ mixand. Step 4 is like step 3, except that the narrowed Gaussian has zero covariance so that the mean of the new $p_b(\boldsymbol{x})$ mixand exactly equals the mean of the original $p_{a'}(\boldsymbol{x})$ mixand. This situation occurs when the original mixand already has a small enough covariance, one whose square-root information matrix already satisfies the LMI in Eq. (6).

Step 5 executes when the importance sampling algorithm calls for more than one sample from a $p_{a'}(\boldsymbol{x})$ mixand whose original square-root information matrix satisfies the LMI in Eq. (6). The algorithm could be redesigned to ignore this situation, in which case distribution $p_b(\boldsymbol{x})$ would contain multiple identical mixands because the exact same mean and square-root information matrix would be assigned in multiple iterations of, respectively, steps 4 and 6. Step 5 circumvents this inefficiency by increasing the weight of an existing $p_b(\boldsymbol{x})$ mixand instead of creating an identical new mixand. This strategy reduces computational cost for any Gaussian mixture filter that uses this resampling method.

Step 8 tests for termination by determining whether the number of attempts to add new mixands is equal to $N_{b\text{target}}$. The algorithm terminates when the total number of iterations of steps 2–8 equals $N_{b\text{target}}$ because the sum of the unnormalized weights w_{unbl} in the step 8 test increases by one for each iteration of these steps. This happens because one and only one of the unnormalized weights gets incremented by one during each iteration. The value of j undergoes a net increment of one for each iteration of steps 2–8 if and only if one or both of the following conditions holds true: $n_{dYai} > 0$ or $l_0(i) = 0$. Otherwise, step 5 is executed as part of the iteration, and the net increment to j is zero.

The index j keeps track of the distribution $p_b(\boldsymbol{x})$ mixand that is in the process of being created. Step 5 decrements this index in recognition of the fact that no new mixand is created if the new mixand would have been identical to an existing one. At the end of the algorithm in step 9, N_b is set equal to j and is guaranteed to be less than or equal to $N_{b\text{target}}$. Step 9 normalizes the unnormalized weights to compute the $p_b(\boldsymbol{x})$ distribution's final weights.

D. Discussion of Algorithm

The value $N_{c\text{target}}$ is an upper bound for the number of mixands to consider in the merging calculations of steps 2–4 of the executive algorithm of Sec. V.B. This value must not be too large. Otherwise, the brute-force solution procedure for the optimization problem in Eqs. (31a–31e) could become too expensive computationally. Recall that this procedure is outlined immediately after Eqs. (31a–31e) in Sec. IV.D.

Two parts of the algorithm have the potential to reduce the eventual number of mixands in distribution $p_b(\boldsymbol{x})$. The first is the mixand merging procedure in steps 2–4 of the main algorithm of Sec. V.B. The second is the mixand reweighting operation in step 5 of Sec. V.C, which happens in lieu of adding a redundant new mixand. Suppose that a nonlinear Bayesian filter converges after initial transients to a distribution that is nearly Gaussian and suppose that the converged distribution's covariance respects the upper bound that corresponds to the square-root information matrix LMI in Eq. (6). In this case, steps 2–4 of the summary algorithm are expected to merge mixands that have any appreciable remaining weight. Furthermore, step 5 of the Sec. V.C algorithm is expected to be reached many times for the few mixands with appreciable nonzero weight that will remain after the merging procedure. The net result will be eventual convergence to a resampled Gaussian mixture $p_b(\boldsymbol{x})$ that has relatively few mixands. These techniques have succeeded in reducing the number of resampled mixands in several example applications to nonlinear Gaussian mixture filtering.

E. Demonstration that the Resampled Gaussian Mixture Converges to the Original in the Limit of a Large Number of Mixands

It is possible to demonstrate that $p_b(\boldsymbol{x})$ becomes an arbitrarily good approximation of $p_{a'}(\boldsymbol{x})$ in the limit as N_b becomes arbitrarily large. If ε in Eq. (31e) of Sec. IV.D is set low enough to ensure that $p_{a'}(\boldsymbol{x})$ is a very good approximation of $p_a(\boldsymbol{x})$, then $p_b(\boldsymbol{x})$ will also converge to $p_a(\boldsymbol{x})$ in the limit of large N. The choice of ε is up to the algorithm designer, and the only drawback of choosing a very small ε will be a failure to merge some mixands in distribution $p_a(\boldsymbol{x})$ and, therefore, the failure to reduce the number of mixands in distribution $p_b(\boldsymbol{x})$ as much as one might like. Therefore, given the possibility of using a sufficient number of mixands in $p_b(\boldsymbol{x})$, the following demonstration that $p_b(\boldsymbol{x})$ converges to $p_{a'}(\boldsymbol{x})$ is easily extensible to a demonstration that $p_b(\boldsymbol{x})$ can be made to converge to $p_a(\boldsymbol{x})$.

In the limit of very large N_b, it is obvious that the probability mass function sampling in step 2 of the Sec. V.C algorithm guarantees the following property of the weights of distribution $p_b(\boldsymbol{x})$:

$$w_{a'i} = \sum_{j \in J_{bi}} w_{bj} \quad \text{for } i = 1, \ldots, N_a \quad (36)$$

where J_{bi} is the set of indices of all distribution-$p_b(\boldsymbol{x})$ mixands that have been generated from mixand i of distribution $p_{a'}(\boldsymbol{x})$ in steps 2–7 of the algorithm of Sec. V.C.

Next, consider the Monte Carlo method of selecting the mixand means of distribution $p_b(\boldsymbol{x})$. It is given in steps 3 and 4 of the Sec. V.C algorithm. When coupled with Eq. (36), this Monte Carlo procedure for generating mixand means implies the following: A good approximation of $p_b(\boldsymbol{x})$ in the limit of large N_b can be derived via partial replacement of the resampled distribution's mixand summation with integral equivalents. That is,

$$\begin{aligned} p_b(\boldsymbol{x}) &= \sum_{i=1}^{N_a} \left(\sum_{j \in J_{bi}} w_{bj} \mathcal{N}_{\text{sr}}(\boldsymbol{x}; \boldsymbol{\mu}_{bj}, \tilde{R}_{ai}) \right) \\ &\cong \sum_{i=1}^{N_a} w_{a'i} \int_{-\infty}^{\infty} \mathcal{N}_{\text{sr}}(\boldsymbol{x}; [\boldsymbol{\mu}_{ai} + \delta\tilde{Y}_{ai}\boldsymbol{\eta}_i], \tilde{R}_{ai}) \mathcal{N}_{\text{sr}}(\boldsymbol{\eta}_i; 0, I)\, \mathrm{d}\boldsymbol{\eta}_i \end{aligned} \quad (37)$$

Note that the use of $\tilde{R}_{ai}$ in place of R_{bj} in the first line of Eq. (37) is consistent with step 6 of the algorithm of Sec. V.C.

The weight and integral in the second line of Eq. (37) approximate the summation over all $j \in J_{bi}$ in the first line by virtue of the way that $\boldsymbol{\mu}_{bj}$ is chosen in step 3 or 4 of the Sec. V.C algorithm. The zero-mean, identity-covariance dummy integration variable $\boldsymbol{\eta}_i$ takes the place of the random-number-generated vector $\boldsymbol{\eta}$ in step 3 of the algorithm of Sec. V.C. Recall that its dimension is n_{dYai}. If a particular mixand of distribution $p_{a'}(\boldsymbol{x})$ yields the dimension $n_{dYai} = 0$ because $\tilde{R}_{ai} = R_{ai}$ satisfies the LMI in Eq. (6), then step 4 of the algorithm of Sec. V.C applies rather than step 3. There is no random-number-generated $\boldsymbol{\eta}$ vector in this case, and $\delta\tilde{Y}_{ai}$ is an empty matrix. Nevertheless, the formula in Eq. (37) can be retained through a redefinition of the $\delta\tilde{Y}_{ai}$ matrix to be an n-by-1 matrix of zeros, which redefines n_{dYai} to equal one. This redefinition maintains the needed relationship between R_{ai}, $\tilde{R}_{ai}$, and $\delta\tilde{Y}_{ai}$ that is given by Eq. (13) if one makes the substitutions $R_{bj} = \tilde{R}_{ai}$ and $dY_{aibj} = \delta\tilde{Y}_{ai}$.

The $\boldsymbol{\eta}_i$ integral in Eq. (37) can be evaluated analytically. One way to evaluate it is to employ the unit normalization formula for a standard vector Gaussian distribution. The resulting integral derivation involves several transformations and a significant amount of matrix algebra. Its details have been omitted for the sake of brevity. After integration, Eq. (37) becomes

$$p_b(\boldsymbol{x}) \cong \sum_{i=1}^{N_a} w_{a'i}\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{ai},[\tilde{R}_{ai}^{-1}\tilde{R}_{ai}^{-T} + \delta\tilde{Y}_{ai}\delta\tilde{Y}_{ai}^{T}])$$
$$= \sum_{i=1}^{N_a} w_{a'i}\mathcal{N}_{\mathrm{sr}}(\boldsymbol{x};\boldsymbol{\mu}_{ai},R_{ai}) \tag{38}$$

The last line of this equation is exactly the definition of Gaussian mixture $p_{a'}(\boldsymbol{x})$, thereby demonstrating the convergence of $p_b(\boldsymbol{x})$ to $p_{a'}(\boldsymbol{x})$ in the limit of large N_b.

VI. Examples of Algorithm Performance

A. Fidelity of the Resampled Gaussian Mixture Approximation

The algorithm described in Secs. II–V has been implemented in MATLAB and tested on several problems. Consider the one-dimensional example whose original Gaussian mixture $p_a(\boldsymbol{x})$ and resampled mixture $p_b(\boldsymbol{x})$ are plotted in Fig. 1. The original $p_a(\boldsymbol{x})$ has three components and is plotted in solid black along the horizontal axis. The approximate $p_b(\boldsymbol{x})$ has 5000 components and is plotted as the dash-dotted gray curve along the same axes. The three components of $p_a(\boldsymbol{x})$ are plotted as dotted gray curves. Their respective weights are 0.4410, 0.0687, and 0.4903; their mean values are −1.0106, 0.5077, and 0.6145; and their standard deviations are 0.7462, 0.4165, and 2.0603. The components of $p_b(\boldsymbol{x})$ all have the same standard deviation: 0.20. This is the upper limit imposed by the LMI in Eq. (6). As can be seen from Fig. 1, $p_b(\boldsymbol{x})$ approximates $p_a(\boldsymbol{x})$ very well. The cost of achieving this excellent fit is the need to use 5000 components to construct $p_b(\boldsymbol{x})$. A fit with an N_b count of only 1000 mixands (not shown) has noticeably less accuracy.

B. Propagation of Probability Density Through a Nonlinear Function Using a Resampled Distribution with Narrowed Mixand Covariances

Figure 1 illustrates an important point about why this Gaussian mixture resampling method has been developed. It plots an example nonlinear function $\boldsymbol{f}(\boldsymbol{x})$ as the dash-dotted black curve. This function is a cubic spline that is defined by its node $\boldsymbol{x}$ values, $\boldsymbol{f}$ values, and $\mathrm{d}\boldsymbol{f}/\mathrm{d}\boldsymbol{x}$ values, as per Table 1. The figure also shows the exact propagation of the probability density function $p_a(\boldsymbol{x})$ through $\boldsymbol{f}(\boldsymbol{x})$ to produce the corresponding probability density function for $\boldsymbol{f}$: $p_f(\boldsymbol{f}) = p_a[\boldsymbol{x}(\boldsymbol{f})]/|\partial\boldsymbol{f}/\partial\boldsymbol{x}|$, where $\boldsymbol{x}(\boldsymbol{f})$ represents the function inverse of $\boldsymbol{f}(\boldsymbol{x})$. This probability density is plotted as the solid black distribution that is shown along the left-hand vertical axis (after being moved to have its zero value line up at the horizontal position $\boldsymbol{x} = -12$ and after being scaled down by a factor of 3 to fit well within the figure's horizontal range). Because $\boldsymbol{f}$ is its independent variable, $p_f(\boldsymbol{f})$ is plotted along the vertical $\boldsymbol{f}$ axis. Also plotted on that axis are two approximations of $p_f(\boldsymbol{f})$. The dotted gray curve is the $p_f(\boldsymbol{f})$ that results from performing simple EKF-type propagations through $\boldsymbol{f}(\boldsymbol{x})$ for the three components of $p_a(\boldsymbol{x})$. The dash-dotted gray curve is similar, except that it applies the EKF-type propagations to the 5000 components of $p_b(\boldsymbol{x})$. It is obvious from this plot that the latter approximation is much closer to the truth. It even reproduces the bimodal peaks of the true distribution. Thus, there can be significant benefit in terms of nonlinear filtering accuracy if one reapproximates $p_a(\boldsymbol{x})$ by a Gaussian mixture $p_b(\boldsymbol{x})$ with covariance bounds on each of its components. This benefit occurs even for the case of $N_b = 1000$ mixands that is mentioned at the end of the previous subsection.

Table 1 Spline node values to define example $f(x)$

x spline nodes	f values at nodes	$\mathrm{d}f/\mathrm{d}x$ values at nodes
−15	0	1/150
−10	1/30	1/120
−9	1/20	3/80
−2	1/3	3/80
−1	7/20	1/60
1	23/60	1/30
1.5	13/30	19/180
10	4/3	8/81
15	53/30	13/150

Note that no particular mathematical criterion has been defined to determine the minimum required number of new mixands N_b that achieves good approximation accuracy of $p_a(\boldsymbol{x})$ and $p_f(\boldsymbol{f})$. The numbers $N_b = 1000$ and $N_b = 5000$ are representative values that help to illustrate the potential accuracy of this new reapproximation method.

C. Bayesian Conditional Probability Density Calculations with a Nonlinear Measurement Function

A different calculation is required to illustrate the benefits of using a resampled Gaussian mixture with bounded component covariances when performing the measurement update of a nonlinear filter. Suppose that $p_a(\boldsymbol{x})$ of Fig. 1 is the a priori probability distribution for $\boldsymbol{x}$, and suppose that $\boldsymbol{f}(\boldsymbol{x})$ of Fig. 1 is a nonlinear measurement function rather than a nonlinear dynamic propagation function. Suppose that the measurement model takes the form

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\nu} \tag{39}$$

where $\boldsymbol{y}$ is the measurement and $\boldsymbol{\nu}$ is Gaussian measurement noise with a mean of zero and a covariance of $P_{\nu\nu}$. Then Bayes' rule dictates that the a posteriori probability distribution of $\boldsymbol{x}$ is

$$p_{\mathrm{posterior}}(\boldsymbol{x}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})p_a(\boldsymbol{x})}{\int_{-\infty}^{\infty} p(\boldsymbol{y}|\boldsymbol{x})p_a(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}}$$
$$= C\exp\{-0.5[\boldsymbol{y} - f(\boldsymbol{x})]^T P_{\nu\nu}^{-1}[\boldsymbol{y} - f(\boldsymbol{x})]\}p_a(\boldsymbol{x}) \tag{40}$$

where C is a normalization constant. This posterior distribution can be approximated as a Gaussian sum by using EKF or UKF calculations to do individual updates for each of the Gaussian components followed by reweighting of the components. The reweighting is based on chi-squared statistics of the components' normalized innovations, as in [10].

Figure 2 presents three a posteriori probability density functions for this example. The measurement error covariance is $P_{\nu\nu} = (0.1)^2$. A truth-model simulation generated $\boldsymbol{x}_{\mathrm{true}} = -2.0965$ and $\boldsymbol{y} = 0.2996$. The solid black curve is the true a posteriori probability density. The dash-dotted gray curve is the result of applying multiple-model Gaussian/EKF calculations to the 5000 mixands of Fig. 1's approximate a priori distribution $p_b(\boldsymbol{x})$. The dotted gray curve is similar, except it applies the approximate multiple-model Gaussian/EKF calculations directly to the three elements of the true a priori distribution $p_a(\boldsymbol{x})$. The dash-dotted gray curve is a much better approximation of the solid black curve than is the dotted gray curve. The improvement of the gray dash-dotted curve vs. the gray dotted curve further illustrates the advantage for nonlinear Gaussian mixture

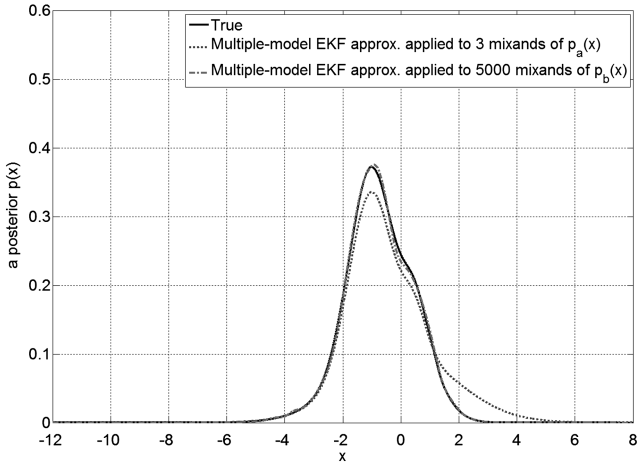

Fig. 2 True and approximate a posteriori probability distributions after a nonlinear measurement update.

filtering when using the mixand covariance limit in this paper's resampling technique.

Although not shown in Fig. 2, significant accuracy improvement occurs even for the case of $N_b = 1000$ mixands that has been mentioned at the ends of the previous two subsections. The decision to plot a case with $N_b = 5000$ mixands has been made to emphasize the possible accuracy of this technique.

It is possible to improve upon the Gaussian mixture reapproximation accuracy and the nonlinear filtering accuracy shown in Figs. 1 and 2 when resampling with N_b as small as 100. Such performance improvements have been achieved by using the more complicated resampling algorithm of [30]. Future research might fruitfully combine ideas from [30] with the present resampling technique to produce a more efficient method.

D. Performance of the Gaussian Mixture Reapproximation Algorithm Within a Bayesian Nonlinear Filter

This paper's new reapproximation algorithm has been used to develop a full nonlinear Gaussian mixture filter. Reference [10] defines this new filtering algorithm and reports the results of applying it to a simulation of the blind tricyclist nonlinear estimation problem of [4].

The blind tricyclist constitutes a challenging seven-state nonlinear estimation problem. It includes unknown planar position and heading states of the tricycle. Its nonlinear relative bearing measurements are made to two moving reference points that are mounted on two separate merry-go-rounds, each with two unknown parameters that must be estimated by the filter. These extra filter states are the angle and the constant angular rate of each merry-go-round.

Two different regularized particle filters have been tried on this problem, one with 3000 particles and another with 10,000 particles. Neither performs very well. The best performance was obtained in [4] using the backward smoothing EKF (BSEKF) of [3], which is also known as the moving-horizon estimator [6]. None of these existing filters exhibited performance that was close to the Cramer–Rao lower bound [4].

The new resampling algorithm of this paper, when embedded in the modified Gaussian mixture filter of [10], achieved better performance than all the filters described in [4]. It used the target number of mixands after resampling $N_{b\text{target}} = 7000$, and it used the following square-root information matrix lower bound in Eq. (6): $R_{\min} = \text{diag}[1/(2.6\text{ m}); 1/(2.6\text{ m}); 1/(1.04\text{ rad}); 1/(0.3467\text{ rad}); 1/(0.4\text{ rad}); 1/(2000\text{ rad/s}); 1/(2000\text{ rad/s})]$. Thus, the position component standard deviations are limited to 2.6 m per axis after dynamic propagation, the heading standard deviation is limited to 1.04 rad (60 deg), the two merry-go-rounds' angular standard deviations are limited to 0.3467 rad (20 deg) and 0.4 rad (23 deg), respectively, and their rate standard deviations are limited to 2000 rad/s (115,000 deg or 318 Hz). The rate standard deviation limits have been set high because the rates do not directly enter any problem function nonlinearities. The component error standard deviations used to define $R_{\min}$ have been selected to make the geometric nonlinearities in the blind tricyclist problem reasonably well approximated by linearized models over the corresponding state uncertainty ranges. The chosen $N_{b\text{target}}$ value enables a reasonably accurate approximation of the wide initial state probability density function using a Gaussian mixture that respects the covariance bounds associated with $R_{\min}$.

The performance of the Gaussian mixture resampling algorithm within the new nonlinear filter is characterized by Figs. 3 and 4. Figure 3 plots the rms position error magnitude time histories for 8 filters along with the Cramer–Rao lower bound for this error. These rms values are computed for 100 Monte Carlo simulations of the estimation problem for each filter for the case of large initial errors. The eight filters include an EKF, two UKFs with different tuning parameters, two BSEKFs with different horizons of explicit backward smoothing (BSEKF A using 30 samples and BSEKF B using 40 samples), two regularized PFs with different particle counts (PF A using 3000 particles and PF B using 10,000 particles), and the new Gaussian mixture filter that includes this paper's resampling algorithm with resampling control parameters defined in the preceding paragraph (designated as the blob filter in Fig. 3). Note that this comparison does not explicitly include comparisons with alternate Gaussian mixture filters (e.g., those of [11–14]). Such a comparison should be done, but it is beyond the scope of the present paper and of [10], which is the source of Fig. 3.

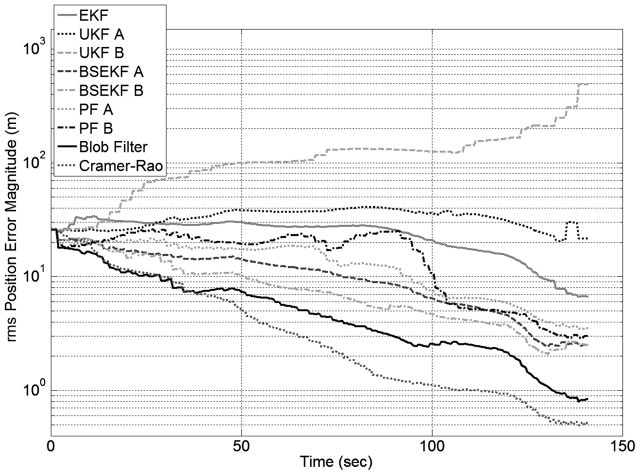

Fig. 3 Blind tricyclist rms position error time histories of eight filters and the Cramer–Rao lower bound, as computed from 100 Monte Carlo simulations (from [10]).

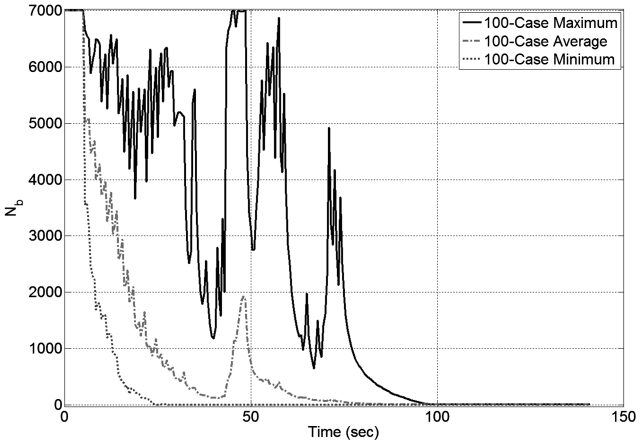

Fig. 4 Maximum, mean, and minimum N_b time histories for blob filter runs on 100 Monte Carlo simulations of the blind tricyclist problem (from [10]).

Figure 3 is similar to Fig. 5 of [4], except that the two PFs show improved performance here in comparison to the figure from [4]. This improved performance results from the removal of an aliasing problem from the PFs that was caused by the 2π ambiguities of the problem's heading angle and merry-go-round angles. These ambiguities, when coupled with the PF's regularization calculations, caused mischief with the original PF results reported in [4].

It is clear from Fig. 3 that the new blob filter achieves the best performance of all eight filters. Its solid black curve is always the lowest of all eight filters, and it is much nearer to the Cramer–Rao lower bound (CRLB) than any of the other filters. At the end of the filtering run, the blob filter rms error is only 60% higher than the CRLB, but the two next best filters, BSEKFs A and B, have final rms errors that are 370% higher than the CRLB.

In fact, the new filter's rms error lies slightly below the CRLB out to $t = 37.5$ s. This result seems inconsistent with the theory that the CRLB really is a lower bound. Furthermore, the curves of four of the other filters lie below the CRLB during the first 9.5 s. These seemingly impossible results are conjectured to be artifacts of using only a finite number of Monte Carlo simulations to generate Fig. 3.

Also considered in [10] is the normalized estimation error squared (NEES) of the filter state as a measure of consistency. The new blob filter exhibits somewhat reasonable consistency, and it displays the second best consistency during the second half of the filtering interval. PF B achieves the best NEES filter consistency.

Another important performance feature of the Gaussian mixture resampling algorithm is depicted in Fig. 4. This figure plots statistics of the actual number of mixands in distribution $p_b(\boldsymbol{x})$ after resampling, N_b. This number varies from case to case for the 100 Monte Carlo runs, and it varies with time. The figure plots the maximum (solid black curve), mean (dash-dotted gray curve), and minimum (dotted gray curve) of N_b over the 100 cases as functions of time since filter initialization. The initial values of all three statistics equal 7000, consistent with the chosen value of $N_{b\text{target}}$. As time progresses through the filtering run, however, all three statistics start to drop. They all drop to very low values by the end of the run at $t = 141$ s, with the final maximum being 8, the mean 3.77, and the minimum 1. This decay of N_b is a useful property because the computational burden of running the blob filter scales linearly with N_b. Any ability to reduce N_b while maintaining filter performance will reduce the filter's need for computational resources.

The sharp drop in all three N_b statistics over time can be attributed to the ad hoc measures by which the resampling algorithm attempts to limit the number of mixands in $p_b(\boldsymbol{x})$. Recall that the first of these measures is the resampler's attempt to merge redundant mixands, as described in Sec. IV and as implemented in steps 2–4 of the executive algorithm defined in Sec. V.B. The second of these measures is the resampler's ability to reweight a single mixand of $p_b(\boldsymbol{x})$ instead of replicating it. This reweighting occurs if the original mixand from distribution $p_a(\boldsymbol{x})$ has a covariance sufficiently small to satisfy the LMI in Eq. (6) along with a weight sufficiently large to be selected multiple times during the probability mass function sampling in step 2 of the algorithm in Sec. V.C.

One might be concerned that the small N_b values near the final time in Fig. 4 could cause problems like those caused by a lack of particle diversity in a PF. This is not the case because the Gaussian mixture blob filter does not rely solely on particle diversity to give width to its approximation of the a posteriori Bayesian distribution. Distribution width is also inherent in the covariance of each mixand. Therefore, a single mixand can have sufficient diversity if the true Bayesian distribution is narrow and nearly Gaussian. Note, also, that it would be possible for a filter based on the present resampling procedure to increase its N_b if an increase were to become necessary. The increase would happen after a dynamic propagation if that propagation added enough state uncertainty to cause the LMI in Eq. (6) to be violated by the mixands that characterized the new a priori filter distribution.

The new blob filter is computationally much more expensive than a simple EKF or UKF. In [4], mean computation times for all of the filters averaged over the 100 Monte Carlo cases were reported. The EKF requires only 0.08 s, on average, to filter the entire data batch when running in MATLAB on a Windows XP Professional Workstation, and the two UKFs require only 1.18 s. BSEKF A requires 60.84 s, on average, and BSEKF B requires 110.6 s. The latter BSEKF requires more execution time because it performs explicit nonlinear smoothing over a longer interval. PF A requires 149 s, and PF B requires 695 s, numbers that differ somewhat from those reported in [4] due to the fix up of the angular aliasing problems. PF B is slower than PF A due to its use of 3.3 times as many particles. The mean execution time for the blob filter is 187 s. Thus, the new blob filter is more expensive computationally than six of the other seven filters, but it uses 3.7 times less computing power than PF B. Better PF performance might be achievable by increasing the number of particles beyond 10,000, but the computational cost would make it much less attractive than the new Gaussian mixture blob filter for this particular problem.

Similar to a PF, the calculations of the blob filter are almost completely parallelizable, all except the inexpensive reweighting at the end of the measurement update [10] and the mixand merging operations of Sec. IV. Therefore, its execution speed has the potential to be significantly increased by mapping it onto a parallel processor.

Most of the details about the blind tricyclist problem and all of its mathematical equations have been omitted from the present discussion for the sake of brevity. The interested reader should consult [4] to learn the details of this benchmark nonlinear estimation problem. A link to MATLAB software for the benchmark problem's various dynamics and measurement functions is cited in [4]. The software can be downloaded by researchers to test their own nonlinear filters.

VII. Conclusions

A new Gaussian mixture reapproximation/resampling algorithm has been developed. It has three goals. First, it seeks to create a new mixture that is a close approximation of the original mixture. Second, it limits the covariances of the elements of the new mixture so that each one will propagate accurately through typical EKF or UKF nonlinear filter calculations, provided that the covariances have been limited to a sufficient degree for a given problem model. The algorithm's third goal, which is of secondary priority, is to limit the number of components of the resampled mixture.

The new resampling algorithm represents a natural generalization of a particle filter's importance resampling, but with new complexities. Covariance matrices of the new mixture components are bounded from above. These bounds define systems of linear matrix inequalities that set lower bounds on the corresponding square-root information matrices. Optimal solutions to the linear matrix inequalities determine new mixand square-root information matrices that are as close as possible to those of the original mixture. Mean values of the new mixture components are sampled from modified components of the original mixture that have reduced covariances. These covariance reductions compensate for the fact that the total covariance of the new mixture is determined by two contributions: one from the reduced covariances of the new mixands and the other from the variability of the new mixands' mean values.

The resampling algorithm has been tested on two sets of example problems. The results show that a good approximation of the original probability density can be achieved with significantly narrowed covariances of the resampled mixands. This achievement enables accurate Bayesian nonlinear estimation calculations via application of simple EKF or UKF operations within the Gaussian mixture framework and using a multiple-model-filter approach. In one Monte Carlo test, the new resampling algorithm enabled a new Gaussian mixture filter to achieve significantly better performance on a difficult seven-state nonlinear estimation problem than has been achieved by four other popular types of nonlinear filters.

References

[1] Julier, S., Uhlmann, J., and Durrant-Whyte, H. F., "New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators," *IEEE Transactions on Automatic Control*, Vol. 45, No. 3, 2000, pp. 477–482.
doi:10.1109/9.847726

[2] Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T., "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing*, Vol. 50, No. 2, Feb. 2002, pp. 174–188. doi:10.1109/78.978374

[3] Psiaki, M. L., "Backward-Smoothing Extended Kalman Filter," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 5, Sept.–Oct. 2005, pp. 885–894. doi:10.2514/1.12108

[4] Psiaki, M. L., "The Blind Tricyclist Problem and a Comparative Study of Nonlinear Filters," *IEEE Control Systems Magazine*, Vol. 33, No. 3, June 2013, pp. 40–54. doi:10.1109/MCS.2013.2249422

[5] Wan, E. A., and van der Merwe, R., "The Unscented Kalman Filter," *Kalman Filtering and Neural Networks*, edited by Haykin, S., Wiley, New York, 2001, pp. 221–280.

[6] Rao, C. V., Rawlings, J. B., and Mayne, D. Q., "Constrained State Estimation for Nonlinear Discrete-Time Systems: Stability and Moving Horizon Approximations," *IEEE Transactions on Automatic Control*, Vol. 48, No. 2, Feb. 2003, pp. 246–258. doi:10.1109/TAC.2002.808470

[7] Psiaki, M. L., "Estimation Using Quaternion Probability Densities on the Unit Hypersphere," *Journal of Astronautical Sciences*, Vol. 54, Nos. 3–4, July–Dec. 2006, pp. 415–431. doi:10.1007/BF03256498

[8] Sorenson, H. W., and Alspach, D. L., "Recursive Bayesian Estimation Using Gaussian Sums," *Automatica*, Vol. 7, No. 4, 1971, pp. 465–479. doi:10.1016/0005-1098(71)90097-5

[9] Alspach, D. L., and Sorenson, H. W., "Nonlinear Bayesian Estimation Using Gaussian Sum Approximations," *IEEE Transactions on Automatic Control*, Vol. 17, No. 4, Aug. 1972, pp. 439–448. doi:10.1109/TAC.1972.1100034

[10] Psiaki, M. L., "The 'Blob' Filter: Gaussian Mixture Nonlinear Filtering with Resampling for Mixand Narrowing," *2014 IEEE/ION Position, Location, and Navigation Symposium*, IEEE, Piscataway, NJ, May 2014, pp. 393–406. doi:10.1109/PLANS.2014.6851397

[11] Van der Merwe, R., and Wan, E., "Gaussian Mixture Sigma-Point Particle Filters for Sequential Probabilistic Inference in Dynamic State-Space Models," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, IEEE, Piscataway, NJ, April 2003, pp. 701–704. doi:10.1109/ICASSP.2003.1201778

[12] Faubel, F., McDonough, J., and Klakow, D., "The Split and Merge Unscented Gaussian Mixture Filter," *IEEE Signal Processing Letters*, Vol. 16, No. 9, Sept. 2009, pp. 786–789. doi:10.1109/LSP.2009.2024859

[13] Huber, M. F., "Adaptive Gaussian Mixture Filter Based on Statistical Linearization," *Proceedings of the 14th International Conference on Information Fusion*, IEEE, Piscataway, NJ, July 2011, pp. 1–8.

[14] Horwood, J. T., and Poore, A. B., "Adaptive Gaussian Sum Filters for Space Surveillance," *IEEE Transactions on Automatic Control*, Vol. 56, No. 8, Aug. 2011, pp. 1777–1790. doi:10.1109/TAC.2011.2142610

[15] Bar-Shalom, Y., Li, X.-R., and Kirubarajan, T., *Estimation with Applications to Tracking and Navigation*, Wiley, New York, 2001, pp. 441–443.

[16] Horwood, J. T., Aragon, N. D., and Poore, A. B., "Gaussian Sum Filters for Space Surveillance: Theory and Simulations," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 6, Nov.–Dec. 2011, pp. 1839–1851. doi:10.2514/1.53793

[17] DeMars, K. J., Bishop, R. H., and Jah, M. K., "Entropy-Based Approach for Uncertainty Propagation of Nonlinear Dynamical Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 4, July–Aug. 2013, pp. 1047–1057. doi:10.2514/1.58987

[18] Bayramoglu, E., Ravn, O., and Andersen, N. A., "A Novel Hypothesis Splitting Method Implementation for Multi-Hypothesis Filters," *Proceedings of the 10th IEEE International Conference on Control & Automation*, IEEE, Piscataway, NJ, June 2013, pp. 574–579.

[19] Havlak, F., and Campbell, M., "Discrete and Continuous Probabilistic Anticipation for Autonomous Robots in Urban Environments," *IEEE Transactions on Robotics*, Vol. 30, No. 2, April 2014, pp. 461–474. doi:10.1109/TRO.2013.2291620

[20] Runnalls, A. R., "Kullback-Leibler Approach to Gaussian Mixture Reduction," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 43, No. 3, July 2007, pp. 989–999. doi:10.1109/TAES.2007.4383588

[21] Terejanu, G., Singla, P., Singh, T., and Scott, P. D., "Uncertainty Propagation for Nonlinear Dynamic Systems Using Gaussian Mixture Models," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 6, Nov.–Dec. 2008, pp. 1623–1633. doi:10.2514/1.36247

[22] Terejanu, G., Singla, P., Singh, T., and Scott, P. D., "Adaptive Gaussian Sum Filter for Nonlinear Bayesian Estimation," *IEEE Transactions on Automatic Control*, Vol. 56, No. 9, Sept. 2011, pp. 2151–2156. doi:10.1109/TAC.2011.2141550

[23] Williams, J. L., and Maybeck, P. S., "Cost-Function-Based Gaussian Mixture Reduction for Target Tracking," *Proceedings of the 6th International Conference on Information Fusion*, IEEE, Piscataway, NJ, 2003, pp. 1047–1054. doi:10.1109/ICIF.2003.177354

[24] Salmond, D. J., "Mixture Reduction Algorithms for Uncertain Tracking," Royal Aerospace Establishment, Technical Rept. 88004, Farnborough, England, U.K., Jan. 1988.

[25] Schoenberg, J. R., Campbell, M., and Miller, I., "Posterior Representation with a Multi-Modal Likelihood Using the Gaussian Sum Filter for Localization in a Known Map," *Journal of Field Robotics*, Vol. 29, No. 2, March–April 2012, pp. 240–257. doi:10.1002/rob.20430

[26] Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic Press, New York, 1981, pp. 37–40, 164.

[27] Kailath, T., *Linear Systems*, Prentice–Hall, Upper Saddle River, NJ, 1980, p. 656.

[28] Psiaki, M. L., "Global Magnetometer-Based Spacecraft Attitude and Rate Estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 2, March–April 2004, pp. 240–250. doi:10.2514/1.1039

[29] Bunea, F., Tsybakov, A. B., Wegkamp, M. H., and Barbu, A., "Spades and Mixture Models," *Annals of Statistics*, Vol. 38, No. 4, Aug. 2010, pp. 2525–2558. doi:10.1214/09-AOS790

[30] Psiaki, M. L., Schoenberg, J. R., and Miller, I. T., "Gaussian Mixture Approximation by Another Gaussian Mixture for 'Blob' Filter Resampling," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2010-7747, Aug. 2010.