The "Blob" Filter: Gaussian Mixture Nonlinear Filtering with Re-Sampling for Mixand Narrowing

Mark L. Psiaki
Sibley School of Mechanical & Aerospace Engineering
Cornell University
Ithaca, N.Y. 14853-7501

Abstract—A new Gaussian mixture filter has been developed, one that uses a re-sampling step in order to limit the covariances of its individual Gaussian components. The new filter has been designed to produce accurate solutions of difficult nonlinear/non-Bayesian estimation problems. It uses static multiple-model filter calculations and Extended Kalman Filter (EKF) approximations for each Gaussian mixand in order to perform dynamic propagation and measurement update. The re-sampling step uses a newly designed algorithm that employs linear matrix inequalities in order to bound each mixand's covariance. Re-sampling occurs between the dynamic propagation and the measurement update in order to ensure bounded covariance in both of these operations. The resulting filter has been tested on a difficult 7-state nonlinear filtering problem. It achieves significantly better accuracy than a simple EKF, an Unscented Kalman Filter, a Moving-Horizon Estimator/Backwards-Smoothing EKF, and a regularized Particle Filter.

Keywords—Kalman Filter; Bayesian Filter; Gaussian Mixture Filter.

I. INTRODUCTION

Kalman filters have been applied to Bayesian estimation problems with stochastic dynamic models and noisy measurements for over 5 decades [1,2]. The original formulation solved the discrete-time problem with linear dynamics, linear measurements, and Gaussian noise and priors [1]. The general discrete-time nonlinear problem, however, has no known closed-form solution. A number of approximate solutions for non-linear/non-Gaussian problems have been developed, and they have met with varying degrees of success. These include the Extended Kalman Filter (EKF) [3,4,5] – perhaps better characterized as the family of EKFs [6], the Unscented or Sigma-Points Kalman Filter (UKF) [7,8], the family of filters known as Particle Filters (PFs) [5,9], the Moving Horizon Estimator [10] -- aka the Backward-Smoothing Extended Kalman Filter (BSEKF) [11], and a family known as Gaussian mixture filters [12,13,14,15,16].

There are two general ways in which these approximate filters can perform poorly. They can diverge, or they may yield sub-optimal or in-consistent estimation accuracy. These two failure modes are illustrated in [17] via simulation tests that use a benchmark 7-state nonlinear estimation problem, the Blind Tricyclist. The UKF can diverge, and its accuracy is very poor. The EKF and PF sometimes nearly diverge, and their accuracies are far from the Cramer-Rao lower bound. The BSEKF converges and has the best estimation accuracy, but its accuracy is not very close to the Cramer-Rao lower bound. Furthermore, the BSEKF is expensive in terms of needed computational resources, and the PF is even more expensive.

Reference [17] did not attempt to include a Gaussian mixture filter in its performance comparison due a known problem of such filters: If their component mixands have covariances that are too large, then the EKF or UKF component calculations will not have sufficient accuracy to achieve good performance [14,15,18,19].

This paper's main contribution is to develop a new type of Gaussian mixture filter that avoids the problem of having covariances which are too large. Its algorithm uses multiple-model filter calculations that are based on first-order EKF approximations carried out on a mixand-by-mixand basis. The measurement update involves re-weighting of the mixands using the usual innovations-based technique of static multiple-model filters [4]. In addition to these standard Gaussian mixture filter operations, a mixture re-sampling/re-approximation step is added between the dynamic propagation and the measurement update. This re-sampling step uses the Gaussian mixture re-approximation algorithm of [20]. This latter algorithm starts with an input Gaussian mixture and a Linear Matrix Inequality (LMI) upper bound that will apply to the covariance matrices of the output mixands. If all of the original mixands respect this bound, then the output mixture essentially equals the input mixture, except that importance-type re-sampling properties will act to eliminate mixands with very low weights. If any of the original mixands do not respect the LMI covariance upper bound, then the re-sampling algorithm constructs a new Gaussian mixture that approximates the original mixture accurately while enforcing the LMI bound. The user selects an upper bound which is small enough to ensure that first-order EKF measurement update and dynamic propagation operations can be applied to each mixand with very little truncation error. This constraint ensures the overall accuracy with which the Gaussian mixture filter approximates the exact Bayesian posterior distribution.

This approach is akin to a PF, but with two important distinctions. First, its re-sampling algorithm approximates one Gaussian mixture by constructing another rather than approximating one ensemble of particles by constructing

Preprint, IEEE/ION PLANS 2014, 5-8 May 2014, Monterey, CA.

another. Second, the re-sampling operation occurs in a different place than for a typical PF, before the measurement update rather than after it [5,9]. This alternate position enables a single re-sampling step to enforce the filter's LMI covariance bound for both the measurement update of the current sample and the dynamic propagation of the next sample.

This new Gaussian mixture filter differs from existing Gaussian mixture filters in several respects. The filters described in [12] and [16] include neither a re-sampling step nor any other means to limit mixand covariances. The filter in [13] includes re-sampling, but not for the purpose of enforcing a covariance upper limit on each mixand. Instead, re-sampling is applied primarily to limit the number of mixands, which is somewhat analogous to the action of importance re-sampling in a PF to prune away particles with low weight. References [14], [15], [18], and [19] develop filters or parts of filters that incorporate Gaussian mixture re-approximation with the goal of limiting the mixand covariances. All of these methods, however, rely on re-approximations of wide 1-dimensional Gaussian elements by a pre-computed set of narrower weighted 1-dimensional mixands. Re-approximation of a multi-dimensional Gaussian mixture is performed using products of mixture re-approximations of 1-dimensional Gaussians along principal axes of each original mixand's covariance matrix. This technique has two drawbacks. First, the number of needed new mixands in the re-approximation is exponential in the problem dimension. Second, there is no obvious way to exploit for the overlap of original wide mixands in order to conserve on the number of narrowed mixands needed in the re-approximated distribution.

Another contribution of this paper is a new square-root information filter (SRIF) formulation of the static multiple-model filter calculations. These calculations form the basis of its Gaussian mixture filter. Versions of the needed calculations have been published in multiple places for covariance filter implementations, e.g., [4], but this paper gives the first known SRIF formulation.

Another contribution is a simulation-based evaluation of the new Gaussian mixture filter on the Blind Tricyclist nonlinear estimation problem [17]. Other Gaussian mixture papers evaluate their proposed filters on problems of much lower dimension, typically 2 or 3, or they evaluate only a single prediction step. The present work tests the new filter on the 7-state Blind Tricyclist problem and compares its performance and computational cost to those of an EKF, two UKFs, two BSEKFs, and two PFs. The new filter's accuracy is also compared to the problem's Cramer-Rao lower bound.

This paper develops and evaluates its new Gaussian mixture Bayesian filter in five main sections. Section II poses the discrete-time nonlinear/non-Gaussian dynamic filtering problem that will be solved using a Gaussian mixture filter. It also reviews the problem's theoretical Bayesian solution. Section III defines a Gaussian mixture using square-root information matrix notation. Section IV defines an LMI that bounds the element covariances of a re-sampled Gaussian mixture, and it reviews the associated new algorithm for Gaussian mixture re-sampling that is presented in [20]. Section V develops the full Gaussian mixture filter algorithm that uses this new re-sampling algorithm. Section VI performs Monte-Carlo simulation tests of the new algorithm's performance on the Blind Tricyclist problem, and it compares this performance with that of other filters. Section VII gives a summary of this paper's new developments along with its conclusions.

II. Discrete-Time Nonlinear Filtering Problem Definition and Theoretical Solution

A. Filtering Problem Definition

The discrete-time nonlinear/non-Gaussian Bayesian filtering problem includes a dynamics model and a measurement model. They take the respective forms:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_k(\boldsymbol{x}_k,\boldsymbol{w}_k) \quad \text{for } k = 0, 1, 2, \ldots \tag{1a}$$

$$\boldsymbol{y}_{k+1} = \boldsymbol{h}_{k+1}(\boldsymbol{x}_{k+1}) + \boldsymbol{v}_{k+1} \quad \text{for } k = 0, 1, 2, \ldots \tag{1b}$$

where k is the discrete-time sample index, $\boldsymbol{x}_k$ is the n_x-dimensional state vector at sample k, $\boldsymbol{w}_k$ is the n_w-dimensional process noise vector that applies for the state transition from sample k to sample k+1, $\boldsymbol{y}_{k+1}$ is the n_y-dimensional measurement vector at sample k+1, and $\boldsymbol{v}_{k+1}$ is the n_y-dimensional measurement noise vector at sample k+1. The vector function $\boldsymbol{f}_k(\boldsymbol{x}_k,\boldsymbol{w}_k)$ is the nonlinear discrete-time dynamics state transition function, and the vector function $\boldsymbol{h}_{k+1}(\boldsymbol{x}_{k+1})$ is the nonlinear measurement function. These two functions are assumed to be continuous with continuous first derivatives.

The definition of 3 *a priori* probability density functions completes the filtering problem definition. The first is the *a priori* probability density for $\boldsymbol{x}_0$. Let it be defined as $p_{x0}(\boldsymbol{x}_0)$. The second is the *a priori* probability density function for each $\boldsymbol{w}_k$. Its definition is $p_{wk}(\boldsymbol{w}_k)$. The third is the *a priori* probability density function for each $\boldsymbol{v}_{k+1}$, which is defined as $p_{vk+1}(\boldsymbol{v}_{k+1})$. Let the probability density functions $p_{wk}(\boldsymbol{w}_k)$ and $p_{vk+1}(\boldsymbol{v}_{k+1})$ be defined for all $k = 0, 1, 2, \ldots$

B. Theoretical Bayesian Solution

The theoretical solution to the filtering problem is the conditional probability density function $p_{xk}(\boldsymbol{x}_k|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)$. This is the probability density of $\boldsymbol{x}_k$ conditioned on all the data starting from the initial measurement $\boldsymbol{y}_1$ and extending up to the measurement $\boldsymbol{y}_k$, which is the last one available at the applicable sample time of $\boldsymbol{x}_k$. This is commonly known as the *a posteriori* probability density function. Let this function be written in short-hand form as $p_{xk}(\boldsymbol{x}_k|k)$, and let the definition of this latter function be extended to include $p_{x0}(\boldsymbol{x}_0|0) = p_{x0}(\boldsymbol{x}_0)$.

An additional useful probability density function is the *a priori* state probability density, $p_{xk+1}(\boldsymbol{x}_{k+1}|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)$. Let this latter function be designated by the short-hand form $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$.

The Bayesian filtering operations start at sample $k = 0$ and successively compute $p_{x1}(\boldsymbol{x}_1|0)$, $p_{x1}(\boldsymbol{x}_1|1)$, $p_{x2}(\boldsymbol{x}_2|1)$, $p_{x2}(\boldsymbol{x}_2|2)$, $p_{x3}(\boldsymbol{x}_3|2)$, $p_{x3}(\boldsymbol{x}_3|3)$, ... These computations can be completely characterized by a recursion that starts with $p_{xk}(\boldsymbol{x}_k|k)$, $p_{wk}(\boldsymbol{w}_k)$, and $p_{vk+1}(\boldsymbol{v}_{k+1})$ and that uses the dynamics and measurement models in (1a) and (1b) to compute $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ followed by $p_{xk+1}(\boldsymbol{x}_{k+1}|k+1)$. These computations start by forming the following probability density functions:

$$p_{xk+1}(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k,\boldsymbol{w}_k) = \delta[\boldsymbol{x}_{k+1} - \boldsymbol{f}_k(\boldsymbol{x}_k,\boldsymbol{w}_k)] \tag{2a}$$

$$\begin{aligned} &p_{xk+1xkwk}(\boldsymbol{x}_{k+1},\boldsymbol{x}_k,\boldsymbol{w}_k|k) \\ &\quad = p_{xk+1}(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k,\boldsymbol{w}_k)p_{xkwk}(\boldsymbol{x}_k,\boldsymbol{w}_k|k) \\ &\quad = \delta[\boldsymbol{x}_{k+1} - \boldsymbol{f}_k(\boldsymbol{x}_k,\boldsymbol{w}_k)]p_{xk}(\boldsymbol{x}_k|k)p_{wk}(\boldsymbol{w}_k) \end{aligned} \tag{2b}$$

where $\delta[]$ is a Dirac delta function that takes a vector argument which is a perturbation of the $\boldsymbol{x}_{k+1}$ vector away from the output of the dynamics model function $\boldsymbol{f}_k(\boldsymbol{x}_k,\boldsymbol{w}_k)$.

The $\boldsymbol{x}_k$ and $\boldsymbol{w}_k$ dependence of the probability density function in (2b) is integrated out to yield the desired *a priori* probability density function at sample $k+1$:

$$\begin{aligned} &p_{xk+1}(\boldsymbol{x}_{k+1}|k) \\ &\quad = \int_{-\infty}^{\infty} d\boldsymbol{w}_k \int_{-\infty}^{\infty} d\boldsymbol{x}_k\, p_{xk+1xkwk}(\boldsymbol{x}_{k+1},\boldsymbol{x}_k,\boldsymbol{w}_k|k) \\ &\quad = \int_{-\infty}^{\infty} d\boldsymbol{w}_k\, p_{wk}(\boldsymbol{w}_k) \int_{-\infty}^{\infty} d\boldsymbol{x}_k \{\delta[\boldsymbol{x}_{k+1} - \boldsymbol{f}_k(\boldsymbol{x}_k,\boldsymbol{w}_k)]p_{xk}(\boldsymbol{x}_k|k)\} \\ &\quad = \int_{-\infty}^{\infty} d\boldsymbol{w}_k\, p_{wk}(\boldsymbol{w}_k)\, p_{xk}[\boldsymbol{f}_k^{-1}(\boldsymbol{x}_{k+1},\boldsymbol{w}_k)\,|\,k]\left[\frac{1}{|\det(\boldsymbol{\Phi}_k)|}\right] \end{aligned} \tag{3}$$

where the function $\boldsymbol{f}_k^{-1}(\boldsymbol{x}_{k+1},\boldsymbol{w}_k)$ in the last line of (3) is the inverse of the dynamics function in (1a). It is defined so that $\boldsymbol{x}_{k+1} = \boldsymbol{f}_k(\boldsymbol{x}_k,\boldsymbol{w}_k)$ and $\boldsymbol{x}_k = \boldsymbol{f}_k^{-1}(\boldsymbol{x}_{k+1},\boldsymbol{w}_k)$ are equivalent, which is the same as saying that the latter equation is valid for all combinations of $\boldsymbol{x}_{k+1}$, $\boldsymbol{x}_k$, and $\boldsymbol{w}_k$ that obey the former equation. The scalar function det() returns the determinant of its matrix input argument. The n_x-by-n_x matrix $\boldsymbol{\Phi}_k$ in (3) is the Jacobian first partial derivative of the dynamics function:

$$\boldsymbol{\Phi}_k = \left.\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{x}_k}\right|_{\{\boldsymbol{f}_k^{-1}[\boldsymbol{x}_{k+1},\boldsymbol{w}_k],\boldsymbol{w}_k\}} \tag{4}$$

The final operations of the Bayesian recursion require the measurement probability density conditioned on the state at sample $k+1$:

$$p_{yk+1}(\boldsymbol{y}_{k+1}|\boldsymbol{x}_{k+1}) = p_{vk+1}[\boldsymbol{y}_{k+1} - \boldsymbol{h}_{k+1}(\boldsymbol{x}_{k+1})] \tag{5}$$

The measurement update is then

$$\begin{aligned} &p_{xk+1}(\boldsymbol{x}_{k+1}|k+1) \\ &\quad = \frac{p_{yk+1}(\boldsymbol{y}_{k+1}|\boldsymbol{x}_{k+1})\,p_{xk+1}(\boldsymbol{x}_{k+1}|k)}{\int_{-\infty}^{\infty} d\breve{\boldsymbol{x}}_k\, p_{yk+1}(\boldsymbol{y}_{k+1}|\breve{\boldsymbol{x}}_{k+1})\,p_{xk+1}(\breve{\boldsymbol{x}}_{k+1}|k)} \\ &\quad = \frac{p_{vk+1}[\boldsymbol{y}_{k+1} - \boldsymbol{h}_{k+1}(\boldsymbol{x}_{k+1})]\,p_{xk+1}(\boldsymbol{x}_{k+1}|k)}{\int_{-\infty}^{\infty} d\breve{\boldsymbol{x}}_k\, p_{vk+1}[\boldsymbol{y}_{k+1} - \boldsymbol{h}_{k+1}(\breve{\boldsymbol{x}}_{k+1})]\,p_{xk+1}(\breve{\boldsymbol{x}}_{k+1}|k)} \end{aligned} \tag{6}$$

Unfortunately, many of the theoretical calculations given in the dynamic propagation in (3) given in the measurement update in (6) cannot be evaluated in closed form. The inverse dynamics function $\boldsymbol{f}_k^{-1}(\boldsymbol{x}_{k+1},\boldsymbol{w}_k)$ often cannot be derived analytically. The last line of (3) is unlikely to allow closed-form evaluation of its integral with respect to $\boldsymbol{w}_k$. The integral in the denominator of the second line of (6) would likely not allow close-form computation. Even if one were able to produce closed-form results for these steps, the resulting $p_{xk+1}(\boldsymbol{x}_{k+1}|k+1)$ likely would be too complicated to allow closed-form computation of the *a posteriori* expected value of $\boldsymbol{x}_{k+1}$ or its covariance. Furthermore, the resulting $p_{xk+1}(\boldsymbol{x}_{k+1}|k+1)$ would constitute an input to the (3) $\boldsymbol{w}_k$ integral for the next sample interval. In all but some special cases, one eventually loses the ability to derive closed-form expressions for the probability density functions of interest.

These difficulties necessitate the development of approximate solutions to this nonlinear/non-Gaussian Bayesian estimation problem. The present approach uses Gaussian mixtures to approximate all of these probability density functions. Given mixtures with individual elements that have sufficiently small covariances, local linearized approximations of the functions $\boldsymbol{f}_k^{-1}(\boldsymbol{x}_{k+1},\boldsymbol{w}_k)$ and $\boldsymbol{h}_{k+1}(\boldsymbol{x}_{k+1})$ can be used. All of the needed integrals are computable in closed form when working with Gaussian mixture probability density functions and local linearizations of the problem model functions.

III. Definition of Gaussian Mixture Probability Density Functions

A. Gaussian Mixture in Square-Root Information Form

A Gaussian mixture probability density function is a weighted sum of individual Gaussian terms. A Gaussian mixture can be written in the following square-root information form

$$p(\boldsymbol{x}) = \sum_{i=1}^{N} w_i \mathcal{N}_{SR}(\boldsymbol{x};\boldsymbol{\mu}_i,R_i) \tag{7}$$

where the scalars w_i for $i = 1, ..., N$ are the mixand weights and the functions

$$\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_i,R_i)=\frac{|\det(R_i)|}{(2\pi)^{n_x/2}}\,e^{-0.5[R_i(\boldsymbol{x}-\boldsymbol{\mu}_i)]^{\mathrm{T}}[R_i(\boldsymbol{x}-\boldsymbol{\mu}_i)]} \quad \text{for } i=1,\ldots,N \tag{8}$$

are individual Gaussian distributions for the n_x-dimensional vector $\boldsymbol{x}$. The i^{th} Gaussian mixand is defined by its n_x-dimensional mean value vector $\boldsymbol{\mu}_i$ along with its n_x-by-n_x square-root information matrix R_i. The corresponding n_x-by-n_x covariance matrix of the i^{th} mixand is $P_i = R_i^{-1}R_i^{-\mathrm{T}}$, with $()^{-\mathrm{T}}$ indicating the transpose of the inverse of the given matrix.

The scalar weights of the distribution must be non-negative, and they must sum to 1. Thus, they obey the conditions:

$$w_i \geq 0 \quad \text{for } i=1,\ldots,N \quad \text{and} \quad 1=\sum_{i=1}^{N} w_i \tag{9}$$

Gaussian mixtures inherit an important property from their Gaussian components: It is straight-forward to calculate various useful integrals that involve Gaussian mixture probability density functions. For example, it is easy to prove unit-normalization of the integral of a Gaussian mixture by using the unit normalization of each individual Gaussian mixand function $\mathcal{N}_{sr}(\boldsymbol{x};\boldsymbol{\mu}_i,R_i)$ along with the unit normalization summation constraint on the weights w_i, which is given in (9). Explicit calculations of the mean and covariance of a Gaussian mixture are carried out using similar techniques. The results are:

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

$$P_{mixture}=\sum_{i=1}^{N} w_i[R_i^{-1}R_i^{-\mathrm{T}}+(\boldsymbol{\mu}_i-\boldsymbol{\mu}_{mixture})(\boldsymbol{\mu}_i-\boldsymbol{\mu}_{mixture})^{\mathrm{T}}] \tag{10}$$

B. Specific Gaussian Mixtures Used to Develop the "Blob" Filter

Five different Gaussian mixture probability density functions are needed in order to develop this paper's Gaussian mixture "blob" filter. One approximates the *a posteriori* state probability density function $p_{xk}(\boldsymbol{x}_k|k)$. Two are needed to approximate the *a priori* state probability density function $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$, one immediately after the dynamic propagation calculations and a second one after the Gaussian mixture re-sampling operation that bounds mixand covariances. A fourth Gaussian mixture is needed to approximate the *a priori* process noise probability density function $p_{wk}(\boldsymbol{w}_k)$, and the fifth approximates the *a priori* measurement noise probability density $p_{vk+1}(\boldsymbol{v}_{k+1})$.

Let these approximations be defined as:

$$p_{xk}(\boldsymbol{x}_k|\,k)=\sum_{i=1}^{\hat{N}_{xk}} \hat{w}_{xki}\mathcal{N}_{sr}(\boldsymbol{x}_k;\hat{\boldsymbol{\mu}}_{xki},\hat{R}_{xxki}) \tag{11a}$$

$$p_{xk+1}(\boldsymbol{x}_{k+1}|\,k)=\sum_{i=1}^{\overline{N}_{xk+1}} \overline{w}_{x(k+1)i}\mathcal{N}_{sr}(\boldsymbol{x}_{k+1};\overline{\boldsymbol{\mu}}_{x(k+1)i},\overline{R}_{xx(k+1)i}) \tag{11b}$$

$$\widetilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)=\sum_{i=1}^{\widetilde{N}_{xk+1}} \widetilde{w}_{x(k+1)i}\mathcal{N}_{sr}(\boldsymbol{x}_{k+1};\widetilde{\boldsymbol{\mu}}_{x(k+1)i},\widetilde{R}_{xx(k+1)i}) \tag{11c}$$

$$p_{wk}(\boldsymbol{w}_k)=\sum_{i=1}^{N_{wk}} w_{wki}\mathcal{N}_{sr}(\boldsymbol{w}_k;\boldsymbol{\mu}_{wki},R_{wwki}) \tag{11d}$$

$$p_{vk+1}(\boldsymbol{v}_{k+1})=\sum_{i=1}^{N_{vk+1}} w_{v(k+1)i}\mathcal{N}_{sr}(\boldsymbol{v}_{k+1};\boldsymbol{\mu}_{v(k+1)i},R_{vv(k+1)i}) \tag{11e}$$

where each of these five Gaussian mixtures is characterized by its number of mixands and by its sets of mixand weights, mean values, and square-root information matrices. The five Gaussian mixtures' respective mixand counts are $\hat{N}_{xk}$, $\overline{N}_{xk+1}$, $\widetilde{N}_{xk+1}$, N_{wk}, and N_{vk+1}. Their respective i^{th} elements have the weights $\hat{w}_{xki}$, $\overline{w}_{x(k+1)i}$, $\widetilde{w}_{x(k+1)i}$, w_{wki}, and $w_{v(k+1)i}$, the mean values $\hat{\boldsymbol{\mu}}_{xki}$, $\overline{\boldsymbol{\mu}}_{x(k+1)i}$, $\widetilde{\boldsymbol{\mu}}_{x(k+1)i}$, $\boldsymbol{\mu}_{wki}$, and $\boldsymbol{\mu}_{v(k+1)i}$, and the square-root information matrices $\hat{R}_{xxki}$, $\overline{R}_{xx(k+1)i}$, $\widetilde{R}_{xx(k+1)i}$, R_{wwki}, and $R_{vv(k+1)i}$.

The Gaussian mixture $\widetilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ in (11c) ostensibly represents the same distribution as the *a priori* Gaussian distribution $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ in (11b), which is the output of the dynamic propagation calculation prior to re-sampling. The alternate *a priori* probability density function $\widetilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ is an approximation of $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ that is generated by applying the re-sampling algorithm of [20]. The primary goal of this re-sampling algorithm is to enforce LMI upper bounds on the covariance of each new mixand $\widetilde{P}_{xx(k+1)i}=\widetilde{R}^{-1}_{xx(k+1)i}\widetilde{R}^{-\mathrm{T}}_{xx(k+1)i}$. A secondary goal is to reduce the number of mixands so that $\widetilde{N}_{xk+1}$ is as small as possible while $\widetilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ is still a good approximation of $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$.

IV. LMI Covariance Bounds and Gaussian Mixture Re-Sampling

The main new feature of this paper's Gaussian mixture filter is the algorithm that generates the re-sampled Gaussian mixture $\widetilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ from $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$. The re-sampling algorithm is developed in [20]. It enforces LMI upper bounds on the covariances of the individual mixands of the new $\widetilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ distribution. Enforcement of these bounds is the key to successful Bayesian estimation through the application of EKF approximations to each mixand within static multiple-model filter calculations. The present section gives an overview this re-sampling algorithm.

A. LMI Square-Root Information Matrix Bounds and Candidate Square-Root Information Matrices for Re-Sampled Mixands

Given the covariance matrix upper bound P_{max}, a corresponding square-root information matrix R_{min} can be computed using Cholesky factorization followed by matrix inversion so that $R_{min}^{-1} R_{min}^{-\mathrm{T}} = P_{max}$. Given R_{min}, the new mixands' covariances will be upper-bounded by P_{max}, in the LMI sense, if and only if their square-root information matrices respect the following LMI lower bounds [20]:

$$\widetilde{R}_{xx(k+1)i}^{\mathrm{T}} \widetilde{R}_{xx(k+1)i} \geq R_{min}^{\mathrm{T}} R_{min} \quad \text{for all } i = 1, \ldots, \widetilde{N}_{xk+1} \quad (12)$$

The choice of the covariance upper bound P_{max} is problem-dependent. It must be chosen small enough so that linear approximations of the dynamics and measurement model functions in (1a) and (1b) are reasonably accurate over about a 2-σ range of P_{max}. It may be allowable for the chosen P_{max} to permit large variations in certain subspaces where the problem model functions only have weak nonlinearities. In the limit of subspaces with perfect linearity of the model functions, very large corresponding projections of P_{max} are acceptable, and the resulting filter calculations will behave somewhat like a Rao-Blackwellized particle filter. It may also be allowable to make the choice of P_{max} dependent of the region of $\boldsymbol{x}_k$ space where the mixands are being generated. The most convenient way to implement such a dependence is to choose P_{max} as function of the mean values of the $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ mixands that are used to generate particular new $\widetilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|k)$ mixands.

The re-sampling algorithm generates each new daughter mixand of $\widetilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|k)$ from an original parent mixand of $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$. The daughter mixand inherits its new square-root information matrix from a perturbed version of the square-root information matrix of the parent mixand. Let the perturbed square-root information matrix associated with the i^{th} mixand of $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ be called $\breve{R}_{xx(k+1)i}$. It is chosen to obey the following two LMIs

$$\breve{R}_{xx(k+1)i}^{\mathrm{T}} \breve{R}_{xx(k+1)i} \geq R_{min}^{\mathrm{T}} R_{min} \quad (13a)$$

$$\breve{R}_{xx(k+1)i}^{\mathrm{T}} \breve{R}_{xx(k+1)i} \geq \overline{R}_{xx(k+1)i}^{\mathrm{T}} \overline{R}_{xx(k+1)i} \quad (13b)$$

while minimizing the following weighted-matrix-norm-squared cost function:

$$J(\breve{R}_{xx(k+1)i}) = \mathrm{Trace}(R_{min}^{-\mathrm{T}} \breve{R}_{xx(k+1)i}^{\mathrm{T}} \breve{R}_{xx(k+1)i} R_{min}^{-1}) \quad (14)$$

Thus, $\breve{R}_{xx(k+1)i}$ is the "smallest" square-root information matrix that is no smaller, in the squared LMI sense, than R_{min} or $\overline{R}_{xx(k+1)i}$. The R_{min} LMI lower bound in (13a) enforces the P_{max} maximum covariance bound, and the $\overline{R}_{xx(k+1)i}$ LMI in (13b) ensures that the daughter mixand covariance is also upper-bounded by the covariance of the parent mixand. This second bound is needed in order to ensure that the parent mixand can be approximated by a set of daughter mixands. Minimization of the matrix-norm-squared cost function in (14) causes the daughter mixand covariance to be as large as possible while respecting the two covariance upper bounds associated with the LMIs in (13a) and (13b). By using the largest possible daughter covariance, the re-sampling algorithm is likely to be able to achieve good re-approximation accuracy with the fewest possible daughter mixands per parent mixand.

Section III of [20] details the calculations that solve for the $\breve{R}_{xx(k+1)i}$ which satisfies the LMIs in (13a) and (13b) while minimizing the cost in (14). They involve a sequence of matrix linear algebra calculations that include matrix inversion, singular value decomposition, and orthonormal/upper-triangular (QR) factorization [21].

The needed calculations yield an additional matrix that is of importance to the re-sampling procedure. It is the matrix square-root of the covariance decrement in going from the parent mixand to the daughter mixand, $\delta\breve{Y}_{xx(k+1)i}$. It obeys the covariance decrement relationship

$$\delta\breve{Y}_{xx(k+1)i} \delta\breve{Y}_{xx(k+1)i}^{\mathrm{T}} = \overline{R}_{xx(k+1)i}^{-1} \overline{R}_{xx(k+1)i}^{-\mathrm{T}} - \breve{R}_{xx(k+1)i}^{-1} \breve{R}_{xx(k+1)i}^{-\mathrm{T}} \quad (15)$$

This covariance square-root matrix always has n_x rows, but it may have fewer than n_x columns. In fact, if $\overline{R}_{xx(k+1)i}$ satisfies the covariance-bounding LMI in (13a), then the optimal $\breve{R}_{xx(k+1)i}$ equals $\overline{R}_{xx(k+1)i}$, and $\delta\breve{Y}_{xx(k+1)i}$ can be an empty array or simply an n_x-by-1 matrix of all zeros.

B. Sampling-Based Algorithm to Choose Mixands of New Gaussian Mixture Distribution

The weights, mean values, and square-root information matrices of the re-sampled $\widetilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|k)$ distribution are chosen by sampling from a modified version of the $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ distribution. This modified version is:

$$p_{tempk+1}(\boldsymbol{x}_{k+1}) = \sum_{i=1}^{\overline{N}_{xk+1}} \overline{w}_{x(k+1)i} \mathcal{N}(\boldsymbol{x}_{k+1}; \overline{\boldsymbol{\mu}}_{x(k+1)i}, \delta\breve{Y}_{xx(k+1)i} \delta\breve{Y}_{xx(k+1)i}^{\mathrm{T}}) \quad (16)$$

Its definition uses the standard covariance form of the vector Gaussian distribution

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, P) = \frac{1}{(2\pi)^{n_x/2} \sqrt{|\det(P)|}} e^{-0.5(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}} P^{-1} (\boldsymbol{x}-\boldsymbol{\mu})} \quad (17)$$

Comparison of (11b) with (16) reveals that the only difference between $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ and $p_{tempk+1}(\boldsymbol{x}_{k+1})$ is that the i^{th} mixand of the former has covariance $\overline{R}_{xx(k+1)i}^{-1} \overline{R}_{xx(k+1)i}^{-\mathrm{T}}$, while the

corresponding mixand of the latter has covariance $\delta\breve{Y}_{xx(k+1)i}\delta\breve{Y}^{T}_{xx(k+1)i}$. The latter covariance is less than or equal to the former in the LMI sense.

The re-sampling algorithm picks the means of its new mixands by sampling directly from $p_{tempk+1}(\boldsymbol{x}_{k+1})$. This sampling procedure is accomplished in two steps that use random number generators and standard Monte-Carlo techniques. The first step samples from a 1-dimensional uniform distribution on the range [0,1]. The uniform sample is compared to the cumulative weights in (16) in order to select the particular element of $p_{tempk+1}(\boldsymbol{x}_{k+1})$ that will be sampled to determine the mean of a new $\tilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ mixand. Let the chosen element of the $p_{tempk+1}(\boldsymbol{x}_{k+1})$ distribution be mixand j. The second step determines the mean of the new l^{th} mixand of $\tilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ by using a random number generator to sample a zero-mean, identity-covariance vector Gaussian distribution with dimension equal to the number of columns in $\delta\breve{Y}_{xx(k+1)j}$. Let this random vector be $\boldsymbol{\eta}_{jl}$. The mean value of the new l^{th} mixand of $\tilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ becomes

$$\tilde{\boldsymbol{\mu}}_{x(k+1)l} = \bar{\boldsymbol{\mu}}_{x(k+1)j} + \delta\breve{Y}_{xx(k+1)j}\boldsymbol{\eta}_{jl} \tag{18}$$

The square-root information matrix of the new l^{th} mixand is set equal to the pre-computed value that corresponds to the LMI-bounded version of the covariance of the j^{th} mixand of $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$:

$$\tilde{R}_{xx(k+1)l} \;=\; \breve{R}_{xx(k+1)j} \tag{19}$$

All of the weights for the new mixands are nominally equal-valued, as in standard importance re-sampling in a PF.

The basic re-sampling algorithm given above is augmented in three ways in [20]. First, it starts with a target value for the number of re-sampled mixands, N_{target}. The actual final value $\tilde{N}_{xk+1}$ will be no larger than this target value, and it could be smaller. Second, it may happen that some of the mixands of $p_{tempk+1}(\boldsymbol{x}_{k+1})$ in (16) will have zero covariance, i.e., $\delta\breve{Y}_{xx(k+1)j} = 0$. If such a mixand is re-sampled multiple times by this algorithm, then a simple-minded application of the algorithm will produce multiple identical mixands of the re-sampled $\tilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ with identical weights. Instead, the algorithm combines these mixands into a single re-sampled mixand with an appropriately increased weight. Third, several of the original mixands of $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ may be nearly identical and have $\delta\breve{Y}_{xx(k+1)j}$ values equal to zero. In this case, the re-sampling algorithm of [20] attempts to merge such mixands before executing its re-sampling steps. Merging is carried out if it results in an initial re-approximation of $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ that is not very different from the original $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ as measured in a functional 2-norm sense. These last two features of the algorithm tend to limit the number of mixands in the re-sampled $\tilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$, $\tilde{N}_{xk+1}$. Experience has shown that these features are likely to reduce $\tilde{N}_{xk+1}$ in situations where the underlying nonlinear/non-Gaussian Bayesian filter has converged to a sufficiently accurate solution, one whose total covariance respects the P_{max} bound.

V. Gaussian Mixture Filtering Algorithm

This paper's new Gaussian mixture filter uses the 5 mixtures in (11a)-(11e) and the dynamics and measurement models in (1a) and (1b) to approximate the operations of an exact nonlinear/non-Gaussian Bayesian filter. The dynamic propagation operation uses $p_{xk}(\boldsymbol{x}_k|k)$ from (11a), $p_{wk}(\boldsymbol{w}_k)$ from (11d), and local linearized approximations of the dynamics model in (1a) in order to compute $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ by evaluating the integral on the third line of (3). These calculations are followed by the Gaussian mixture re-sampling algorithm of [20], which has been reviewed in Section IV. It forms $\tilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ to approximate $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ in a way that respects the LMI mixand covariance upper bound expressed by (12). It also attempts to reduce the needed number of mixands. The final filtering step is the measurement update. It uses $\tilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ from (11c), $p_{vk+1}(\boldsymbol{v}_{k+1})$ from (11e), and local linearized approximations of the measurement model (1b) in order to compute $p_{xk+1}(\boldsymbol{x}_{k+1}|k+1)$ by evaluating the second line of (6). The dynamic propagation calculations and the measurement update calculations are standard static multiple-model filter operations [4], except they are carried out using EKF techniques cast in the form of SRIF calculations.

The next two subsections define, respectively, the multiple-model/EKF/SRIF dynamic propagation and the multiple-model/EKF/SRIF measurement update. The final subsection combines these operations with the re-sampling algorithm of [20] and Section IV in order to define this paper's full Gaussian mixture filter. It also discusses the reason for placement of the re-sampling step between the dynamic propagation and the measurement update.

A. Gaussian Mixture Dynamic Propagation using Mixand-by-Mixand EKF Calculations

Multiple-model filter dynamic propagation operations are used to approximate the integral in the last line of (3). The approximation uses EKF calculations for each mixand. The input mixands for this calculation are those of the product distribution:

$$p_{xk}(\boldsymbol{x}_k|\,k)p_{wk}(\boldsymbol{w}_k) = \{\sum_{i=1}^{\hat{N}_{xk}} \hat{w}_{xki}\mathcal{N}_{sr}(\boldsymbol{x}_k;\hat{\boldsymbol{\mu}}_{xki},\hat{R}_{xxki})\}\times \{\sum_{j=1}^{N_{wk}} w_{wkj}\mathcal{N}_{sr}(\boldsymbol{w}_k;\boldsymbol{\mu}_{wkj},R_{wwkj})\} \tag{20}$$

This product causes there to be $\bar{N}_{xk+1} = \hat{N}_{xk}N_{wk}$ mixands in the calculation's $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ output distribution.

The multiple-model dynamic propagation implements the SRIF form of the EKF dynamic propagation steps for each of the $\bar{N}_{xk+1}$ products of *a posteriori* state mixands $i = 1, ..., \hat{N}_{xk}$ and process noise mixands $j = 1, ..., N_{wk}$. The index of each resulting *a priori* state mixand for $\boldsymbol{x}_{k+1}$ is $l = N_{wk}(i\text{-}1) + j$, and the corresponding dynamic propagation calculations for this mixand are:

$$\bar{w}_{x(k+1)l} = \hat{w}_{xki} w_{wkj} \tag{21a}$$

$$\bar{\boldsymbol{\mu}}_{x(k+1)l} = \boldsymbol{f}_k(\hat{\boldsymbol{\mu}}_{xki}, \boldsymbol{\mu}_{wkj}) \tag{21b}$$

$$Q_{kl}\begin{bmatrix} \hat{R}_{wwkl} & \hat{R}_{wx(k+1)l} \\ 0 & \bar{R}_{xx(k+1)l} \end{bmatrix} = \begin{bmatrix} R_{wwkj} & 0 \\ -\hat{R}_{xxki}\Phi_{kl}^{-1}\Gamma_{kl} & \hat{R}_{xxki}\Phi_{kl}^{-1} \end{bmatrix} \tag{21c}$$

where the calculations in (21c) start with the matrix on the right-hand side and perform a QR factorization of that matrix in order to compute the outputs on the left-hand side. These outputs are the orthonormal matrix Q_{kl}, the square, upper-triangular matrices $\hat{R}_{wwkl}$ and $\bar{R}_{xx(k+1)l}$, and the additional matrix $\hat{R}_{wx(k+1)l}$.

The calculations in (21a)-(21c) have been derived by first substituting the linearized approximation

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{f}_k^{-1}(\boldsymbol{x}_{k+1}, \boldsymbol{w}_k) \\ &\cong \hat{\boldsymbol{\mu}}_{xki} + \Phi_{kl}^{-1}\{\boldsymbol{x}_{k+1} - \boldsymbol{f}_k(\hat{\boldsymbol{\mu}}_{xki}, \boldsymbol{\mu}_{wkj}) - \Gamma_{kl}[\boldsymbol{w}_k - \boldsymbol{\mu}_{wkj}]\} \end{aligned} \tag{22}$$

with

$$\Phi_{kl} = \left.\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{x}_k}\right|_{(\hat{\boldsymbol{\mu}}_{xki}, \boldsymbol{\mu}_{wkj})} \text{ and } \Gamma_{kl} = \left.\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{w}_k}\right|_{(\hat{\boldsymbol{\mu}}_{xki}, \boldsymbol{\mu}_{wkj})} \tag{23}$$

into the product term $\hat{w}_{xki}\mathcal{N}_{sr}(\boldsymbol{x}_k; \hat{\boldsymbol{\mu}}_{xki}, \hat{R}_{xxki}) \times w_{wkj}\mathcal{N}_{sr}(\boldsymbol{w}_k; \boldsymbol{\mu}_{wkj}, R_{wwkj})$ of the Gaussian mixture in (20). The resulting approximate term is a joint Gaussian in $\boldsymbol{x}_{k+1}$ and $\boldsymbol{w}_k$. Its integral with respect to $\boldsymbol{w}_k$ in the last line of (3) can be evaluated by using the Gaussian unit normalization condition. The values of $\bar{w}_{x(k+1)l}$, $\bar{\boldsymbol{\mu}}_{x(k+1)l}$, and $\bar{R}_{xx_{x(k+1)l}}$ in (21a)-(21c) have been defined so that the resulting integral equals $\bar{w}_{x(k+1)l}\mathcal{N}_{sr}(\boldsymbol{x}_{k+1}; \bar{\boldsymbol{\mu}}_{x(k+1)l}, \bar{R}_{xx_{x(k+1)l}})$, consistent with standard SRIF calculations [22].

In order for the linearized approximation in (22) to be sufficiently accurate, it is required that the $p_{wk}(\boldsymbol{w}_k)$ mixand covariances $R_{wwkj}^{-1}R_{wwkj}^{-\mathrm{T}}$ for $j = 1, ..., N_{wk}$ be sufficiently small. Thus, the square-root information matrices for the process noise mixands must obey an LMI of the form

$$R_{wwkj}^{\mathrm{T}} R_{wwkj} \geq R_{wmin}^{\mathrm{T}} R_{wmin} \text{ for all } j = 1, ..., N_{wk} \tag{24}$$

where $P_{wmax} = R_{wwmin}^{-1}R_{wwmin}^{-\mathrm{T}}$ is the covariance upper bound for the *a priori* process noise mixands. If the original formulation of the filtering problem does not respect this process noise mixand LMI, then the Gaussian mixture re-sampling algorithm of [20], as reviewed in Section IV of the present paper, must be applied to the original process noise distribution in order to produce the distribution of (11d) that will be used by the filter.

The orthonormal matrix Q_{kl} is discarded after the operations in (21a)-(21c), and the matrices $\hat{R}_{wwkl}$ and $\hat{R}_{wx(k+1)l}$ are also discarded unless one wants to do a backwards smoothing pass after the forwards filtering pass.

The dynamic propagation of the mixand square-root information matrix in (21c) relies on the inverse of the mixand state transition matrix Φ_{kl}. Alternate calculations can be implemented in situations where Φ_{kl} is not invertible provided that the resulting $\bar{R}_{xx(k+1)l}$ will not be infinite, which will be the case for a process noise model that keeps the filtering problem non-singular. One suitable alternate version uses square-root covariance propagation followed by matrix inversion in order to revert back to a square-root information matrix representation of the mixand covariance.

B. Gaussian Mixture Measurement Update using Mixand-by-Mixand EKF Computations

A multiple-model filter measurement update is used in conjunction with EKF techniques in order to approximate the last line of (6). The input mixands for this calculation are those of the product distribution:

$$\begin{aligned} &\tilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)\,p_{vk+1}[\boldsymbol{y}_{k+1} - \boldsymbol{h}_{k+1}(\boldsymbol{x}_{k+1})] = \\ &\left(\sum_{i=1}^{\tilde{N}_{xk+1}} \tilde{w}_{x(k+1)i}\mathcal{N}_{sr}\{\boldsymbol{x}_{k+1}; \tilde{\boldsymbol{\mu}}_{x(k+1)i}, \tilde{R}_{xx(k+1)i}\}\right) \times \\ &\left(\sum_{j=1}^{N_{vk+1}} w_{v(k+1)j}\mathcal{N}_{sr}\{[\boldsymbol{y}_{k+1} - \boldsymbol{h}_{k+1}(\boldsymbol{x}_{k+1})]; \boldsymbol{\mu}_{v(k+1)j}, R_{vv(k+1)j}\}\right) \end{aligned} \tag{25}$$

Thus, there are $\hat{N}_{xk+1} = \tilde{N}_{xk+1}N_{vk+1}$ mixands in the filter's final *a posteriori* distribution $p_{xk+1}(\boldsymbol{x}_{k+1}|k+1)$.

The EKF/SRIF multiple-model measurement update performs the Bayesian calculations in the last line of (6) using local linearizations of the measurement model function $\boldsymbol{h}_{k+1}(\boldsymbol{x}_{k+1})$. The linearized approximation used in the ij^{th} product term of (25) is:

$$\boldsymbol{h}_{k+1}(\boldsymbol{x}_{k+1}) \cong \boldsymbol{h}_{k+1}(\tilde{\boldsymbol{\mu}}_{x(k+1)i}) + H_{(k+1)i}[\boldsymbol{x}_{k+1} - \tilde{\boldsymbol{\mu}}_{x(k+1)i}] \tag{26}$$

where

$$H_{(k+1)i} = \left.\frac{\partial \boldsymbol{h}_{k+1}}{\partial \boldsymbol{x}_{k+1}}\right|_{\tilde{\boldsymbol{\mu}}_{x(k+1)i}} \tag{27}$$

Next, the measurement update re-arranges each mixand product term in (25), $\widetilde{w}_{x(k+1)i}\mathcal{N}_{sr}\{\boldsymbol{x}_{k+1};\widetilde{\boldsymbol{\mu}}_{x(k+1)i},\widetilde{R}_{xx(k+1)i}\}\times w_{\nu(k+1)j}\mathcal{N}_{sr}\{[\boldsymbol{y}_{k+1}-\boldsymbol{h}_{k+1}(\boldsymbol{x}_{k+1})];\boldsymbol{\mu}_{\nu(k+1)j},R_{\nu\nu(k+1)j}\}$, into the equivalent *a posteriori* form $\hat{w}_{unx(k+1)l}\mathcal{N}_{sr}\{\boldsymbol{x}_{k+1};\hat{\boldsymbol{\mu}}_{x(k+1)l},\hat{R}_{xx(k+1)l}\}$. This is done for each of the N_{xk+1} products of *a priori* state mixands i = 1, ..., $\widetilde{N}_{xk+1}$ and measurement noise mixands j = 1, ..., N_{vk+1}. The index of each corresponding *a posteriori* state mixand is $l = N_{vk+1}(i\text{-}1) + j$. Equivalence between the two forms is possible because substitution of the linearized measurement function model from (26) into the measurement noise mixand causes the original product term to be Gaussian in $\boldsymbol{x}_{k+1}$. The following standard SRIF measurement update calculations [22] and multiple-model filter calculations are designed to ensure the equivalence of these two Gaussian forms

$$\hat{Q}_{(k+1)l}\begin{bmatrix}\hat{R}_{xx(k+1)l}\\0\end{bmatrix}=\begin{bmatrix}\widetilde{R}_{xx(k+1)i}\\R_{\nu\nu(k+1)j}H_{(k+1)i}\end{bmatrix} \tag{28a}$$

$$\begin{bmatrix}\hat{\boldsymbol{z}}_{x(k+1)l}\\\boldsymbol{z}_{r(k+1)l}\end{bmatrix}=\hat{Q}^{\mathrm{T}}_{(k+1)l}\begin{bmatrix}0\\R_{\nu\nu(k+1)j}\{\boldsymbol{y}_{k+1}-\boldsymbol{h}_{k+1}(\widetilde{\boldsymbol{\mu}}_{x(k+1)i})-\boldsymbol{\mu}_{\nu(k+1)j}\}\end{bmatrix} \tag{28b}$$

$$\hat{\boldsymbol{\mu}}_{x(k+1)l}=\widetilde{\boldsymbol{\mu}}_{x(k+1)i}+\hat{R}^{-1}_{xx(k+1)l}\hat{\boldsymbol{z}}_{x(k+1)l} \tag{28c}$$

$$\hat{w}_{unx(k+1)l}=\widetilde{w}_{x(k+1)i}w_{\nu(k+1)j}\left(\frac{|\det(\widetilde{R}_{xx(k+1)i})\det(R_{\nu\nu(k+1)j})|}{|\det(\hat{R}_{xx(k+1)l})|(2\pi)^{n_y/2}}\right)\times\exp\{-0.5\boldsymbol{z}^{T}_{r(k+1)l}\boldsymbol{z}_{r(k+1)l}\} \tag{28d}$$

The calculations in (28a) QR factorize the input matrix on the right-hand side in order to compute the orthonormal matrix $\hat{Q}_{(k+1)l}$ and the square, upper-triangular *a posteriori* square-root information matrix $\hat{R}_{xx(k+1)l}$, which both appear on the left-hand side. Equation (28b) forms the normalized measurement residual on the right-hand side and transforms the result using the transpose of $\hat{Q}_{(k+1)l}$ in order to compute the residual error vector $\boldsymbol{z}_{r(k+1)l}$ and the vector $\hat{\boldsymbol{z}}_{x(k+1)l}$. The latter vector is used in (28c) to update the mixand mean.

Equation (28d) computes the un-normalized version of the *a posteriori* mixand weight. Four of the product terms on the right-hand side of (28d) are standard in a static multiple-model re-weighting formula: the two pre-update mixand weights, the power of 2π, and the exponential that involves a sum of squares of normalized measurement residuals [4]. The terms involving determinants of square-root information matrices are required for this new SRIF form of the static multiple-model measurement update. They replace a term involving the determinant of the innovations covariance matrix that appears in the covariance form of this re-weighting formula [4].

The final step of the Gaussian mixture measurement update re-normalizes the new mixand weights. Re-normalization occurs after the calculations in (28a)-(28d) have been carried out for all of the mixands. The re-normalization takes the form:

$$\hat{w}_{x(k+1)l}=\frac{\hat{w}_{unx(k+1)l}}{\sum_{m=1}^{\hat{N}_{xk+1}}\hat{w}_{unx(k+1)m}}\quad\text{for } l=1,\ldots,\hat{N}_{xk+1} \tag{29}$$

This completes the calculations on the last line of (6) because the sum of the unnormalized weights in the denominator of (29) equals the integral of the probability density product in the denominator of (6).

Given that these calculations could involve a large number of mixands, $\hat{N}_{xk+1}$, it is important to carry them out in the most efficient manner possible. One important efficiency is to evaluate the linearization calculations in (26) and (27) only $\widetilde{N}_{xk+1}$ times, not $\hat{N}_{xk+1}$ times. Similarly, the determinant $\det(\widetilde{R}_{xx(k+1)i})$ only needs to be evaluated $\widetilde{N}_{xk+1}$ different times, and the determinant $\det(R_{\nu\nu(k+1)j})$ only requires N_{vk+1} independent evaluations. Furthermore, the usual upper-triangularity of $\widetilde{R}_{xx(k+1)i}$, $R_{\nu\nu(k+1)j}$, and $\hat{R}_{xx(k+1)l}$ expedites the determinant calculations because the determinant of an upper-triangular matrix equals the product of its diagonal elements.

C. Complete Gaussian Mixture Filter Operations

The Gaussian mixture filtering algorithm starts with the following inputs:

a) the *a posteriori* state Gaussian mixture at sample 0, $p_{x0}(\boldsymbol{x}_0|0)$,
b) the process noise and measurement noise Gaussian mixtures $p_{wk}(\boldsymbol{w}_k)$ and $p_{vk+1}(\boldsymbol{v}_{k+1})$ at samples k = 0, 1, 2, ...
c) the square-root information matrix LMI lower bound R_{min} that is used in (12), and
d) the target number of mixands after each state re-sampling N_{target}.

Given these inputs, the approximate nonlinear/non-Gaussian Bayesian calculations of the new Gaussian mixture filter proceed according to the following 7 steps:

1. If necessary, re-sample $p_{x0}(\boldsymbol{x}_0|0)$ using the Gaussian mixture re-sampling algorithm of [20] in order to ensure that the LMI in (12) is satisfied by each mixand's square-root information matrix. If re-sampling is necessary, then use N_{target} as the target/maximum number of re-sampled mixands.
2. Set the current sample index to k = 0.
3. Perform the multiple-model SRIF/EKF dynamic propagation calculations in (21a)-(21c) for all

$\bar{N}_{xk+1} = \hat{N}_{xk} N_{wk}$ mixands of the new *a priori* distribution $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$. Use the matrices $\boldsymbol{\Phi}_{kl}$ and $\boldsymbol{\Gamma}_{kl}$ for l = 1, ..., $\bar{N}_{xk+1}$ from (23) in these calculations.

4. Perform Gaussian mixture re-sampling of $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$ in order to produce $\tilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ with mixand square-root information matrices that all satisfy the LMI in (12). Use the re-sampling procedure of [20], as discussed in this paper's Section IV. Use N_{target} as the target/maximum number of re-sampled mixands.
5. Perform the multiple-model SRIF/EKF measurement update calculations in (28a)-(28d) for all $\hat{N}_{xk+1} = \tilde{N}_{xk+1} N_{vk+1}$ mixands of the new *a posteriori* distribution $p_{xk+1}(\boldsymbol{x}_{k+1}|k+1)$. Use the matrices $H_{(k+1)i}$ for i = 1, ..., $\tilde{N}_{xk+1}$ from (27) in these calculations.
6. Use (29) to compute the normalized mixand weights of the new *a posteriori* distribution $p_{xk+1}(\boldsymbol{x}_{k+1}|k+1)$.
7. Replace k by k+1. Stop if the incremented k value is the last sample of the filtering run. Otherwise, return to Step 3.

This is a recursive procedure. Each iteration of Steps 3-7 produces the *a posteriori* distribution $p_{xk+1}(\boldsymbol{x}_{k+1}|k+1)$ that becomes the input distribution $p_{xk}(\boldsymbol{x}_k|k)$ for the next iteration after k is incremented in Step 7.

The most important purpose of Step 4's mixture re-sampling is to bound the mixand covariances using the LMI in (12). This bounding ensures sufficient accuracy of the EKF linearizations of the dynamics and measurement models in (22) and (26). It is obvious that enforcement of the LMI bound in Step 4 can ensure accuracy of the measurement model linearization in (26) because the measurement update calculations of Step 5 occur immediately after the re-sampling step. Satisfaction of the LMI bound in (12) during the dynamic propagation may seem uncertain because the measurement update in Steps 5 and 6 happens after the re-sampling and before the Step-3 dynamic propagation of the filter recursion's next iteration. Note, however, that the SRIF measurement update calculation in (28a) guarantees that

$$\hat{R}^{\mathrm{T}}_{xx(k+1)l}\hat{R}_{xx(k+1)l} \geq \tilde{R}^{\mathrm{T}}_{xx(k+1)i}\tilde{R}_{xx(k+1)i} \qquad (30)$$

This inequality re-states the fact that the measurement update of a linear filter can never increase the covariance of the corresponding state estimate. The re-sampling in algorithm Step 4 ensures that the LMI in (12) is satisfied by each $\tilde{R}_{xx(k+1)i}$. Equation (30) and the transitivity of LMIs ensure that

$$\hat{R}^{\mathrm{T}}_{xx(k+1)l}\hat{R}_{xx(k+1)l} \geq R^{\mathrm{T}}_{min}R_{min} \text{ for all } l = 1, ..., \hat{N}_{xk+1} \qquad (31)$$

Therefore, the filter need not re-sample after its measurement update. Equation (31) guarantees that the *a posteriori* mixands at the end of the k^{th} iteration of Steps 3-7 are sufficiently narrow to admit accurate EKF dynamic propagation calculations in Step 3 of the k+1$^{\text{st}}$ iteration.

The second goal of the mixture re-sampling in Step 4 is to limit the number of mixands. If possible, this step seeks to reduce this number as much as is practical without adversely affecting the accuracy of the re-sampled distribution. One can show that $\bar{N}_{xk+1} = \tilde{N}_{xk} N_{vk} N_{wk}$. If N_{vk} or N_{wk} is greater than 1, then the number of mixands will grow during execution of Steps 5-7 of the k-1$^{\text{st}}$ iteration of the algorithm and Step 3 of the k^{th} iteration. Without the re-sampling in Step 4, the number of mixands would grow geometrically without bound, and the calculations would quickly become intractable. The re-sampling in Step 4 bounds each $\tilde{N}_{xk}$ to be no greater than N_{target}, thereby bounding the possible growth in the number of mixands. Normally such a bound can be enforced without undue loss of accuracy because many of the mixands "lost" in the re-sampling process have insignificant weights or are redundant with other mixands.

As will be shown in Section VI, the re-sampling algorithm of [20] includes features that have the potential to form a $\tilde{p}_{xk+1}(\boldsymbol{x}_{k+1}|\,k)$ that is an accurate approximation $p_{xk+1}(\boldsymbol{x}_{k+1}|k)$, but with the number of re-sampled mixands $\tilde{N}_{xk+1}$ much less than N_{target}. When this happens, the Gaussian mixture filter has the potential to be very efficient computationally, almost as efficient as a standard EKF or UKF.

The new Gaussian mixture filter differs markedly from a standard PF in its placement of the re-sampling algorithm between the dynamic propagation of Step 3 and the measurement update of Steps 5 and 6. A typical PF performs importance re-sampling after the measurement update [5,9]. The re-positioning of the re-sampling step in the present filter has been driven by its need to enforce the LMI bound in (12). If the re-sampling step took place after the measurement update, then the potential for mixand covariance growth during the dynamic propagation on the next iteration could cause violation of the LMI in (12) at the point of measurement model linearization in (26) and (27). The fact of covariance shrinkage in the measurement update allows the filter to avoid all such problems by placing the re-sampling step between the dynamic propagation and the measurement update. This placement enables each filter iteration's one re-sampling operation to enforce the LMI bound in (12) at two separate points of the iteration.

The only negative consequence of moving the re-sampling step is the potential to retain insignificant mixands too long. The re-weighting calculations of the measurement update in (28d) and (29) can cause some updated mixands to have negligible weights. Typically this happens when a given mixand has unreasonably large normalized measurement residuals in its $\boldsymbol{z}_{r(k+1)l}$ vector from (28b). Such mixands have virtually no impact on the remainder of the filter calculations. Failure to eliminate them through importance re-sampling

after the measurement update results in wasted computation. The wasted operations, however, persist only until re-sampling Step 4 on the next iteration of the algorithm. That step includes mixand importance considerations in its re-sampling calculations, which eliminate all mixands with negligible weights.

VI. Gaussian Mixture Filter Performance on the Blind Tricyclist Problem

Monte-Carlo simulation tests of the new Gaussian mixture filter have been carried out for the Blind Tricyclist nonlinear filtering problem of [17]. Its performance is compared to that of the same nonlinear filters that are evaluated in [17], an EKF, two UKFs, two Moving Horizon Estimators (also called Backwards-Smoothing EKFs -- BSEKFs), and two PFs.

A. Review of Blind Tricyclist Problem

The Blind Tricyclist problem is developed in [17] as a benchmark nonlinear filtering problem. It involves a blind tricyclist who navigates around an amusement park based on relative bearings measured between his heading and the locations of 2 friends on 2 merry-go-rounds who shout to him intermittently. He knows which friend is on which merry-go-round and can distinguish their voices. He also knows the location and diameter of each merry-go-round, but he does not know each friend's initial phase angle on the merry-go-round (i.e., which horse the friend rides), nor does he know the constant rate of rotation of each merry-go-round. He seeks to determine his east-west/north-south position and his heading angle. In order to do this, he also must estimate the "nuisance" parameters that characterize the two friends' merry-go-rounds' instantaneous angles and constant rates. Thus, his state vector $\boldsymbol{x}_k$ has 7 elements, his position coordinates X_k and Y_k, his heading angle θ_k, the phase angles of the two friends' merry-go-round positions ϕ_{1k} and ϕ_{2k}, and their merry-go-rounds' constant rates $\dot{\phi}_{1k}$ and $\dot{\phi}_{2k}$. The k subscript on the latter two quantities is somewhat misleading in that their true values do not change with time, but their estimates can change with time as a result of measurement updates.

The Blind Tricyclist uses a kinematic model that includes known input time histories for the steer angle and the speed. The process noise includes small white-noise errors in these two commanded input time histories along with wheel slippage that allows slight violations of the tricycle's planar rolling constraints. The dynamics models for the merry-go-rounds are kinematic and have no uncertainty beyond their unknown initial angles and rates.

Relative bearing measurements to each merry-go-round-riding friend are made once every 3 seconds, with the two friends shouting out of phase with each other. During the 141 second duration of the considered filtering run, there are a total of 47 relative bearing measurements made to each friend. The filter sample intervals are only 0.5 seconds long. Thus, there are three sample times with no measurement data between each sample time that has bearing data. The filter algorithm skips measurement update Steps 5 and 6 in the absence of bearing data.

Figure 1 illustrates this estimation problem. It shows a top view of the amusement park with the two merry-go-rounds and a simulated true tricyclist trajectory for a particular case. Also shown are various filters' estimates of this trajectory. The merry-go-rounds lie slightly to the left of center, with a smaller one to the north drawn in green and a larger one to the south drawn in magenta. The truth tricyclist trajectory is shown in solid blue. The tricyclist starts at the blue asterisk position to the south-east of the figure's center. He proceeds almost due north until he takes a ~90° right turn. At the end of this first turn he stops and then executes a ~90° left turn traveling in reverse. At the end of this second turn he stops again at the northern-most point of his trajectory. He is now facing south. Next, he proceeds south until he is almost due east of the mid-point between the two merry-go-rounds. He now executes a second ~90° right turn and heads almost due west until he finishes his travels at a point between the two merry-go-rounds.

Many additional details about the Blind Tricyclist are contained in [17]. They include the equations that define the dynamics function $\boldsymbol{f}_k(\boldsymbol{x}_k,\boldsymbol{w}_k)$ and the measurement function

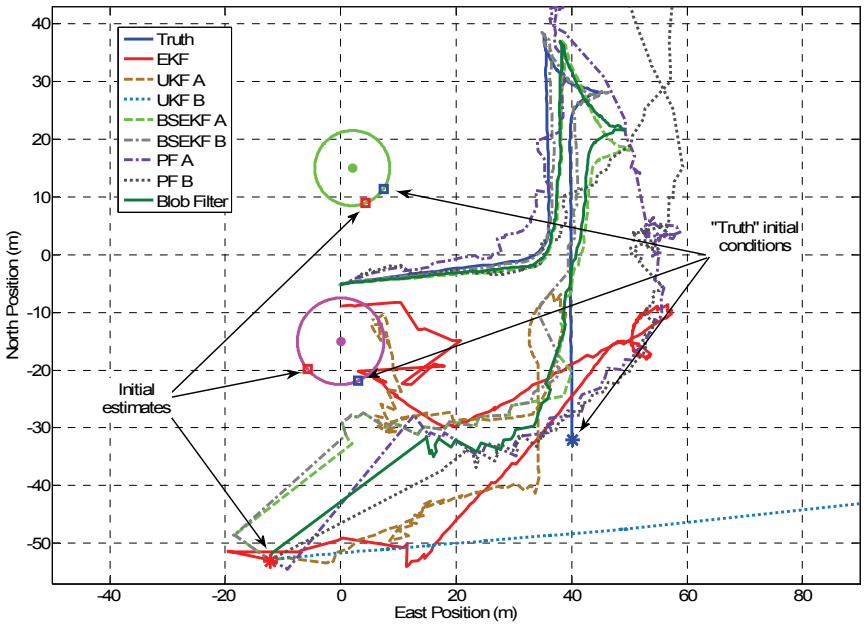

Fig. 1. Plan view of the blind tricyclist problem showing the two merry-go-rounds, the simulated truth position trajectory (blue solid curve), and the corresponding estimated trajectories for 8 different filters.

$\boldsymbol{h}_{k+1}(\boldsymbol{x}_{k+1})$, the process-noise and measurement-noise covariances, the tricycle geometry, the merry-go-round geometry, and the initial estimation error covariance. Two problems are considered, one with a small initial covariance that allows a simple EKF to converge reasonably well, and another with a much larger initial uncertainty that causes severe convergence problems for a number of the filters. Only the difficult latter case is considered here. Reference [17] cites a link to MATLAB code that includes all Blind Tricyclist model functions, a truth-model simulation, and a first-order EKF solution. That link is included here as [23].

B. Comparison of the Gaussian Mixture Blob Filter with other Nonlinear Filters for a Representative Blind Tricyclist Case

The Gaussian mixture filter has been applied to a truth-model simulation of the Blind Tricyclist problem. This implementation uses $N_{wk} = 1$ mixand for the process noise and $N_{vk+1} = 1$ mixand for the measurement noise, thereby modeling these as Gaussian white-noise sequences. There is no need to re-sample the process noise 1-element Gaussian "mixture" in order to respect the bound on its covariance associated with (24) because its original covariance already respects a sufficiently small bound.

The Gaussian mixture filter's re-sampling algorithm has been tuned for the Blind Tricyclist problem as follows: The value of N_{target} is set to 7,000 re-sampled mixands, though results will show that the filter is able to reduce the number of mixands far below this value after initial convergence has been achieved. The LMI lower-bound on the state mixand square-root information matrices in (12) is enforced using the value R_{min} = diag[1/(2.6 m); 1/(2.6 m); 1/(1.04 rad); 1/(0.3467 rad); 1/(0.4 rad); 1/(2000 rad/sec); 1/(2000 rad/sec)]. The first two values correspond to maximum east-west and north-south position error standard deviations of 2.6 m for each mixand. The third term implements a maximum heading standard deviation per mixand of 1.04 rad (60 deg). The fourth and fifth terms specify each mixand's maximum angular standard deviations for the two merry-go-round phase angles, 0.3467 rad (20 deg) for the southern merry-go-round and 0.4 rad (23 deg) for the northern one. The last two terms are per-mixand merry-go-round rate standard-deviation limits of 2,000 rad/sec (318 Hz or 1.15×10^5 deg/sec). These rate limits are intentionally very large because the rates only enter linearly into the dynamics function $\boldsymbol{f}_k(\boldsymbol{x}_k,\boldsymbol{w}_k)$ and not at all into the measurement function $\boldsymbol{h}_{k+1}(\boldsymbol{x}_{k+1})$. Their nonlinear effects are indirect, through the merry-go-round phase angle terms. The mixand covariance limits on the phase angles suffice to limit the indirect nonlinear effects of large merry-go-round rate variances.

Figure 1 plots the east-west and north-south tracking performance of the Gaussian mixture filter on an example simulated problem. The initial position estimate of the filter is given by the red asterisk in the lower left-hand part of the figure. This is obviously far from the initial truth position designated by the blue asterisk in the bottom half of the figure and somewhat right of center. Also shown in the figure are the estimated position time histories of 7 other filters that have been tested in [17]. In fact, Fig. 1 corresponds to the case that produced Fig. 4 of [17]. Therefore, most of the other filters' estimated trajectories in Fig. 1 are identical to those of [17]'s Fig. 4.

For the two PF's, however, the trajectories in Fig. 1 differ from those of [17]. During the course of the present research, it was discovered that PFs and Gaussian mixture filters can be confused by the 2π cycle ambiguity of the tricycle heading angle θ_k and the merry-go-round phase angles ϕ_{1k} and ϕ_{2k}. In the Gaussian mixture filter, this aliasing-like effect is benign. It only impacts the filter's computation of the overall state mean and error covariance, as per (10). If this mean and covariance are required as outputs, then one can remove this aliasing effect by performing a 2π relative unwrapping operation between the mean heading angles and the mean merry-go-round phase angles of the filter's individual mixands. For the PFs, however, failure to compensate for this effect during filter operation can have a dramatic impact on the regularization process that has been used to avoid collapse of particle diversity. It is necessary to do inter-particle unwrapping of these angles after each measurement update before regularized re-sampling occurs. The results reported in [17] do not include this type of PF data processing. The results reported here include it. Therefore, the PF results in the present paper exhibit dramatic improvements in comparison to those of [17].

Even with the improved PF results, the best three filters in Fig. 1 are BSEKF A (light green dashed curve), BSEKF B (grey dash-dotted curve), and the new Gaussian mixture Blob Filter (dark green solid curve). These are the only filters that track near the truth blue solid curve during its initial north-bound leg. They are the only filters that show the initial two turns with the stops at the end of each and with the backing-up maneuver performed during the second turn. The EKF (red solid curve) and UKF A (brown dashed curve) never produce anything like the true trajectory, though they do manage to stay within the general area of tricycle motion. UKF B (cyan dotted curve) diverges out of the figure's field of view and never returns. Its sigma points spread is tuned somewhat differently than that of UKF A [17]. PF A (purple dash-dotted curve) uses 3,000 particles, and PF B (dark grey dotted curve) uses 10,000. Neither of them reproduces the stopping or left-turning back-up maneuver, and PF B even exits the figure's field of view temporarily. In their favor, both PFs eventually approach the true trajectory and the estimated trajectories of the 3 good filters on the final westward leg. The best performance in Fig. 1 appears to be that of BSEKF B, which employs explicit nonlinear smoothing for the 40 half-second sample/propagation intervals that precede the filter sample of interest. The next best performance appears to be that of the new Gaussian mixture Blob Filter, and BSEKF A has the 3rd best performance. BSEKF A employs explicit nonlinear smoothing over a shorter window, only 30 half-second sample/propagation intervals. The performance of the new Blob Filter is only slightly better than that of BSEKF A, and this better performance occurs primarily during the first right-ward turn, the subsequent stop, and the initial part of the backwards left-hand turn.

C. *Statistical Filter Analysis using 100 Monte-Carlo Simulation Cases*

A Monte-Carlo study has been conducted using 100 different truth-model simulations. Each simulation had the same nominal starting point for the tricyclist and the same nominal steer and speed time history as produced the blue solid curve in Fig. 1. Each truth trajectory varied somewhat from that in Fig. 1 due to process noise. The filter's initial distribution $p_{x0}(\boldsymbol{x}_0|0)$ was a Gaussian with the same large covariance for each case, but with a different mean value for each of the 100 Monte-Carlo runs. The distribution of mean values was chosen to cause the 100-case statistics of the actual errors between the truth initial state and each $p_{x0}(\boldsymbol{x}_0|0)$ mean to be consistent with the common covariance of the 100 $p_{x0}(\boldsymbol{x}_0|0)$ distributions.

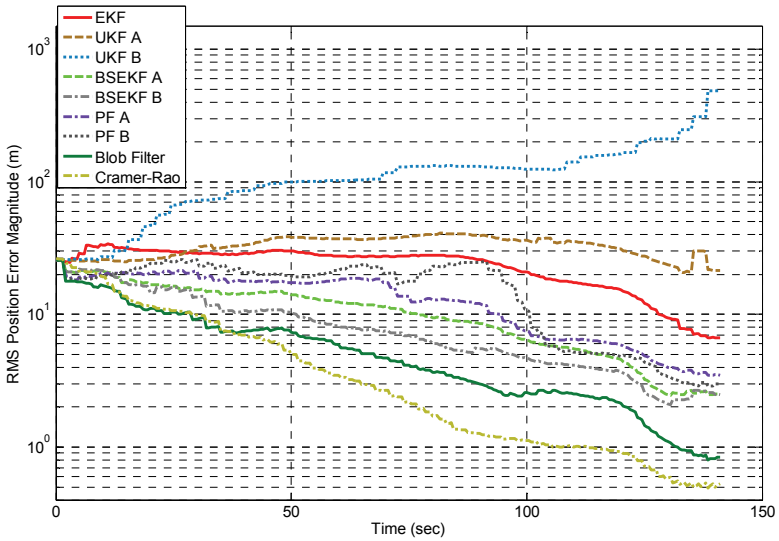

Fig. 2. Blind tricyclist RMS position error magnitude time histories for 8 different filters and the corresponding Cramer-Rao lower bound, as computed using a 100-case Monte-Carlo simulation.

Figures 2 and 3 present summary statistical results from these simulations. They compare the new Gaussian mixture Blob Filter with the 7 other filters that have been used to generate Fig. 1. Figure 2 plots the root-mean-square (RMS) east-west/north-south position error magnitude time history for each filter, as averaged over the 100 Monte-Carlo cases. It also plots the Cramer-Rao lower bound for this error as approximated by averaging over the 100 simulations. The better filter is the one with the lowest RMS error in Fig. 2. Figure 3 plots the fraction of the 100 cases in which the normalized square of the state error $(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k)^{\mathrm{T}} \hat{P}_{xxk}^{-1} (\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k)$ exceeds the 99.99% probability threshold of a degree-7 χ^2 distribution. In this calculation, $\hat{\boldsymbol{x}}_k$ and $\hat{P}_{xxk}$ are, respectively, the mean and covariance of a given filter's approximation of the *a posteriori* probability density function $p_{xk}(\boldsymbol{x}_k|k)$. The fraction of cases that violate this limit should only be about 0.0001 if the state estimation error is nearly Gaussian. Therefore, a consistent filter will produce a curve at or very near 0 in Fig. 3. Note that Figs. 2 and 3 are nearly identical to Figs. 5 and 6 of [17], except that PF A and PF B show better performance than in [17] due to the fixed 2π ambiguity problem noted earlier.

It is obvious from Fig. 2 that the new Gaussian mixture Blob Filter performs significantly better than any of the other filters. Its RMS position error time history (dark green solid curve) is consistently the lowest, and it is much closer to the Cramer-Rao lower bound (tan/green dash-dotted curve) than that of the next best filter, BSEKF B (dark grey dash-dotted curve). Thus, the superior performance of BSEKF B in Fig. 1 appears to be an outlier, a statistical exception to the rule of Fig. 2 that the Blob filter achieves the best RMS accuracy throughout the trajectory.

PF B (grey dotted curve) displays the best the filter consistency in Fig. 3. Its curve is always very close to zero. The new Gaussian mixture Blob Filter (dark green solid curve) has the second best consistency after t = 70 seconds, but one or two other filters display better consistency in the early stages of the filtering interval, UKF A (brown dashed curve) and UKF B (cyan dotted curve). Given that PF B, UKF A, and UKF B are significantly less accurate than the Blob Filter, as demonstrated in Fig. 2, their superior consistency is not very useful. The two filters with accuracy closest to that of the Blob filter in Fig. 2 are BSEKF A (light green dashed curve) and BSEKF B (grey dash-dotted curve). In Fig. 3, however, both of them display significantly poorer consistency than the Blob filter. In summary, the new

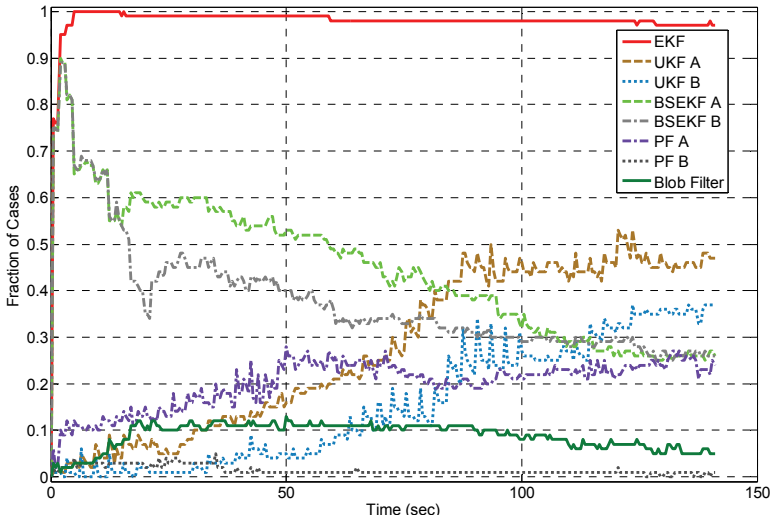

Fig. 3. Blind tricyclist consistency tests for 8 different filters that show the percentage of normalized squared state errors that lie above the 99.99% threshold of a degree-7 χ^2 statistic, as computed using 100 Monte-Carlo simulation cases.

Gaussian mixture Blob filter achieves the best accuracy for this problem, and it's consistency is better than 4 of the other filters for the entire filtering interval and better than 6 of the 7 other filters during the second half of the interval.

Figure 4 investigates the mixand count performance of the Gaussian mixture filter for the 100 Monte-Carlo simulation cases. It plots max($\hat{N}_k$) vs. t_k (blue solid curve), mean($\hat{N}_k$) vs. t_k (red dash-dotted curve), and min($\hat{N}_k$) vs. t_k (light green dashed curve), where the maximum, mean, and minimum are computed over the 100 Monte-Carlo cases. These curves all start at the value 7,000, which is the N_{target} value that has been used for the re-sampling in Step 4 of Subsection V.C. Eventually, the re-sampling algorithm is able to reduce this number through its two ad hoc strategies for developing an accurate Gaussian mixture re-approximation with a reduced number of mixands [20]. By the end of the filter run, the maximum $\hat{N}_k$ is 8, the mean is 3.77, and the minimum is 1.

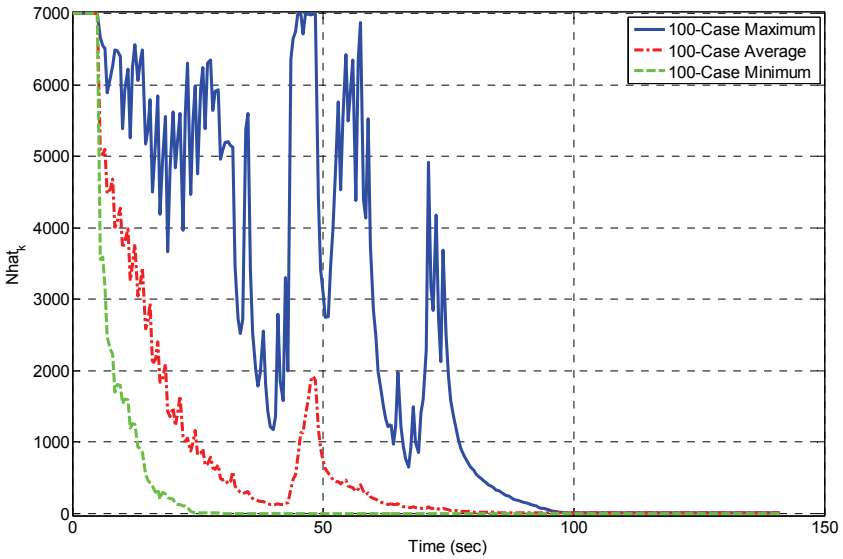

Fig. 4. Time histories of the maximum, mean, and minimum statistics of the Gaussian mixture filter's a posteriori mixand count $\hat{N}_k$ for 100 Monte-Carlo simulation cases.

In a PF context, the low terminal values of $\hat{N}_k$ would be interpreted as a collapse of particle diversity and would indicate filter failure. In the Gaussian mixture context, the filter can function well with a very small number of mixands if the true *a posteriori* probability density function is well modeled by the resulting mixture. Clearly this is the case for the present example, as indicated by the excellent accuracy results given in Fig. 2.

Note that the Gaussian mixture filter has the ability to increase $\hat{N}_k$ should an increase be needed. This ability relies on a coupling of two processes. The first process is the expansion of the mixand covariances of the state probability distribution that can occur during dynamic propagation. The second process is the enforcement of the LMI in (12) during Gaussian mixture re-sampling. A filter with few mixands could experience mixand covariance expansion during the dynamic propagation to the point of violating the LMI in (12) after the propagation. The re-sampling procedure would split the expanded mixands into multiple mixands. This expansion and splitting procedure could continue for multiple samples and eventually result in a large number of mixands. This scenario has occurred in simulation tests of the Gaussian mixture blob filter on a different estimation problem.

One last filter performance metric to consider is the needed computational resources. The average time to run each filter over the 141 second data interval has been computed, with averaging carried out over the 100 Monte-Carlo cases. These averages have been computed when running the filter algorithms on a 3 GHz Windows XP Professional Workstation using MATLAB code. The results are as follows: EKF 0.08 sec, UKFs A and B 1.18 sec, BSEKF A 60.84 sec, BSEKF B 110.6 sec, PF A 149 sec, PF B 695 sec, and the new Gaussian mixture Blob Filter 187 sec. Thus, the Blob Filter is the second most expensive filter, but it uses only 27% as much computation time, on average, as does the 10,000-particle PF B. Note, also, that the Blob Filter is only 70% more expensive computationally than the next most accurate filter, BSEKF B. Furthermore, the Blob Filter would have a more favorable average time comparison for longer filter runs because its largest computational costs occur during the convergence from the large initial errors, when it needs to use many mixands in order to accurately approximate the underlying probability density functions, as documented in Fig. 4.

The new Blob Filter is highly parallelizable. The only calculations that require "communication" between mixands are the weight normalization at the end of the measurement update in (29) and the parts of the re-sampling algorithm that merge mixands. Therefore, greatly increased execution speed could be achieved by mapping the algorithm onto a parallel processor.

VII. SUMMARY AND CONCLUSIONS

A new Gaussian mixture nonlinear filter has been developed. It uses EKF calculations to implement a static multiple-model filter in which each of its Gaussian mixands constitutes a model. These calculations are implemented in SRIF form. The key new element of this filter is its re-sampling algorithm, which executes between the dynamic propagation step and the measurement update step. The primary goal of the re-sampling algorithm is to produce an accurate approximation of the original *a priori* distribution while enforcing an LMI upper bound on the covariance of each of its mixands. If this bound is tuned properly for a given problem, then it ensures that the multiple-model EKF

computations will accurately approximate the underlying Bayesian calculations of an exact nonlinear/non-Gaussian filter. The re-sampling step also limits the number of mixands in the distribution, and it can reduce the number of mixands significantly when many of them would otherwise be redundant.

This new filter can be interpreted as generalizing the concept of a Particle Filter. Its generalization uses "fattened" components -- Gaussian mixands with finite covariances -- instead of particles that have infinitesimal covariances. Therefore, its components might reasonably be designated as "blobs", and the overall filter might reasonably be called a "Blob Filter."

The new Blob Filter has been tested on a difficult 7-state benchmark nonlinear filtering problem, the Blind Tricyclist problem. Monte-Carlo simulation tests demonstrate that the new filter is more accurate than a number of other filters, including an EKF, two UKFs, two Moving-Horizon Estimators/BSEKFs, and two regularized Particle Filters. Its accuracy is significantly closer to the Cramer-Rao lower bound than that of the two next best filters, the BSEKFs. Its speed of execution is slow, but it requires only 27% as much computational time as the most expensive filter considered in this study, a 10,000-particle PF. The new filter's consistency between its computed and actual estimation error covariance is imperfect, but not nearly as imperfect as the two next best filters. Given the Blob Filter's superior accuracy, its bounded computational costs, and its reasonable consistency, it represents a good candidate for solution of difficult nonlinear filtering problems.

REFERENCES

[1] Kalman, R.E., "A New Approach to Linear Filtering and Prediction Problems," *Trans. ASME Journal of Basic Engineering*, Vol. 82, March 1960, pp. 34-45.

[2] Kalman, R.E., and Bucy, R.S., "New Results in Linear Filtering and Prediction Theory," *Trans. ASME Journal of Basic Engineering*, Vol. 83, March 1961, pp. 95-108.

[3] Brown, R.G., and Hwang, P.Y.C., *Introduction to Random Signals and Applied Kalman Filtering, 3rd Edition*, J. Wiley & Sons, (New York, 1997), pp. 345-346.

[4] Bar-Shalom, Y., Li, X.-R., and Kirubarajan, T., *Estimation with Applications to Tracking and Navigation*, J. Wiley & Sons, (New York, 2001), pp. 92-98, 381-395, 441-466.

[5] Ristic, B., Arulampalam, S., and Gordon, N., *Beyond the Kalman Filter*, Artech House, (Boston, 2004), pp. 19-22, 24-31, 35-60, 94-98.

[6] Daum, F., "Nonlinear filters: Beyond the Kalman filter," *IEEE Aerospace & Electronic Systems Magazine*, Vol. 20, No. 8, August 2005, pp. 57–69.

[7] Wan, E.A., and van der Merwe, R., "The Unscented Kalman Filter," *Kalman Filtering and Neural Networks*, S. Haykin, ed., J. Wiley & Sons, (New York, 2001), pp. 221-280.

[8] Julier, S., Uhlmann, J., and Durrant-Whyte, H.F., "A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators," *IEEE Trans. on Automatic Control*, Vol. AC-45, No. 3, 2000, pp. 477-482.

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

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

[11] Psiaki, M.L., "Backward-Smoothing Extended Kalman Filter," *Journal of Guidance, Control, & Dynamics*, Vol. 28, No. 5, Sept.-Oct. 2005, pp. 885-894.

[12] Sorenson, H.W., and Alspach, D.L., "Recursive Bayesian Estimation Using Gaussian Sums," *Automatica*, Vol. 7, No. 4, 1971, pp. 465-479.

[13] 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, & Signal Processing*, (Hong Kong), IEEE, Apr. 2003. Available at http://www.cse.ogi.edu/~rudmerwe/pubs/index.html.

[14] Horwood, J.T., and Poore, A.B., "Adaptive Gaussian Sum Filters for Space Surveillance," *IEEE Trans. on Automatic Control*, Vol. 56, Issue 8, Aug. 2011, pp. 1777–1790.

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

[16] Terejanu, G., Singla, P., Singh, T., and Scott, P., "Adaptive Gaussian Sum Filter for Nonlinear Bayesian Estimation," *IEEE Trans. on Automatic Control*, Vol. 56, Issue 9, Sept. 2011, pp. 2151–2156.

[17] 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.

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

[19] Bayramoglu, E., Ravn, O., and Andersen, N.A., "A Novel Hypothesis Splitting Method Implementation for Multi-Hypothesis Filters," *Proc. 10th IEEE International Conference on Control & Automation*, Hangzhou, China, June 12-14, 2013, pp. 574-579.

[20] Psiaki, M.L., Schoenberg J.R., and Miller, I.T., "Gaussian Sum Re-Approximation for use in a Nonlinear Filter," submitted to the *Journal of Guidance, Control, & Dynamics*, Jan. 2014. Available online at http://gps.mae.cornell.edu/psiaki_etal_gaussianmixresamp_jgcd2014submission.pdf.

[21] Gill, P.E., Murray, W., and Wright, M.H., Practical *Optimization*, Academic Press, (New York, 1981), pp. 37-40.

[22] Bierman, G.J., *Factorization Methods for Discrete Sequential Estimation*, Academic Press, (New York, 1977), pp. 69-76, 115-122.

[23] Psiaki, M.L., "Blind Tricyclist Problem MATLAB Functions, Example Input/Output Data File, and Example Use in EKF Calculations," Cornell University, Ithaca, New York, available online at http://gps.mae.cornell.edu/blind_tricyclist_models_simulation_example.zip, Jan. 2013.