Kernel-Based Statistical EnGMF

Dalton Durant
Dept. of Aerospace Engineering
and Engineering Mechanics
The University of Texas at Austin
Austin, TX, USA
ddurant@utexas.edu

Renato Zanetti
Dept. of Aerospace Engineering
and Engineering Mechanics
The University of Texas at Austin
Austin, TX, USA
renato@utexas.edu

***Abstract*—In an effort to create fast and reliable nonlinear filters for the cislunar regime, this work presents a kernel-based Statistical Ensemble Gaussian Mixture Filter (SEnGMF). The SEnGMF performs kernel density estimation (KDE) on the joint state-measurement distribution, transforming it into a Gaussian mixture. From this, the necessary statistical moments are computed to obtain an estimate for the posterior density. This process avoids linearizing the measurement model, computing Jacobians, and inverting individual innovation matrices, thereby improving computational efficiency with respect to the EnGMF. Results for a cislunar relative navigation problem show that the proposed SEnGMF's accuracy and consistency are on par with the EnGMF's, all while having noticeably faster simulations times.**

***Index Terms*—state estimation, nonlinear filtering, cislunar navigation**

I. Introduction

The Ensemble Gaussian Mixture Filter (EnGMF) is a nonlinear filter introduced in [1]. Its key feature involves transforming a prior ensemble of independently and identically distributed (i.i.d.) particles into a Gaussian mixture model (GMM), effectively producing a smooth and accurate approximation of the prior probability density function (PDF). The original approach in [1] uses kernel density estimation (KDE) to achieve the transformation; however, since its inception, a number of approaches have emerged that try to tackle this transformation differently, *e.g.,* [2]–[8].

Of particular interest is the statistical approach presented in [8], where rather than transforming the prior, this method uses a clustering algorithm on the joint state-measurement distribution to cluster like-particles, transforming the joint into a GMM representation. From this GMM of the joint, they use the conditioning property of Gaussian mixtures to then obtain an estimate for the posterior density in the update step. This approach statistically computes its moments, hence skipping any need to linearize a measurement function and evaluate Jacobians for every mixture component. Though, it still requires inverting an innovation matrix for each component (just like in the EnGMF), which can be costly in large amounts. But even then, the computational cost of computing individual innovation matrices pales in comparison to performing clustering. Clustering algorithms like expectation maximization (EM) [9], [10] and K-means [11] are computationally complex and typically require good initializations [12], [13]. Therefore, the statistical approach in [8] does not lend itself to being fast for practical applications like onboard computers [14].

Instead, we propose a kernel-based Statistical EnGMF (SEnGMF) that avoids clustering altogether by performing an efficient KDE technique on the joint state-measurement distribution, transforming it quickly into a GMM. This process not only avoids clustering, thereby making it more efficient than [8], but also avoids linearizing the measurement model, computing Jacobians, and inverting individual innovation matrices, hence improving computational efficiency relative to the EnGMF. This work tests the filter in a cislunar relative navigation problem, relying solely on sparse angles-only measurements, demonstrating the need for nonlinear filters in the cislunar regime. The results show that, for sufficient ensemble sizes, the proposed kernel-based SEnGMF is just as accurate and consistent of a filter as the EnGMF, but achieves noticeably faster simulation times.

This paper continues as follows. Section II gives background to KDE, the EnGMF, and the filter from [8] that uses clustering on the joint distribution. Then, Section III presents the methodology of the proposed kernel-based SEnGMF, detailing its algorithm, benefits, assumptions, and limitations to consider. Next, Section IV sets up the cislunar relative navigation problem and tests the proposed filter, discussing its numerical results. Finally, Section V provides a conclusion with future directions.

II. Background

A. Kernel Density Estimator

For a Monte Carlo (MC) approach, one way to approximate a PDF is to compute its first two moments via the sample mean and covariance from a set of random samples. This approach is commonly used in linear filters like the Ensemble Kalman Filter (EnKF) [15]–[17], [18, Ch. 6, pp. 153–167], where the PDF can be approximated as a single Gaussian defined by these moments.

But using a Gaussian representation can be seen as partially ignoring one of the main advantages of an MC approach, that is, its ability to represent non-Gaussian probability distributions. Rather than using a single Gaussian to represent the PDF, a standard kernel-based technique can be used instead in which a sum of Gaussian kernels forms a continuous representation from a set of random samples.

Let a set of random samples be represented by $\{\mathcal{D}^{(i)} \in \mathbb{R}^d \colon i = 1, \ldots, n\}$, then the multivariate kernel density estimator with kernel $\mathcal{K}$ and bandwidth β is defined by

$$\hat{g}(\boldsymbol{x}) \coloneqq \frac{1}{n\,\beta^{d/2}} \sum_{i=1}^{n} \mathcal{K}\left(\beta^{-1/2}(\boldsymbol{x} - \mathcal{D}^{(i)})\right), \tag{1}$$

where the kernel function $\mathcal{K}(\cdot)$ is a function defined on d-dimensional $\boldsymbol{x}$, satisfying

$$\int_{\mathbb{R}^d} \mathcal{K}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = 1.$$

It is common to have the kernel be a radially symmetric unimodal PDF[1], for example in the multivariate case:

$$\mathcal{K}(\boldsymbol{x}) = (2\pi)^{-d/2}\,\exp(-\frac{1}{2}\boldsymbol{x}^{\top}\boldsymbol{x}). \tag{2}$$

A noticeable issue is that the bandwidth β gets applied equally in all directions. However, if some directions are disproportionally scaled, then multiplying each by β may have an adverse effect, causing extreme differences of spread in the different directions. A reasonable fix, suggested by [21, Ch. 3, p. 68], is to: first, *pre-whiten* the samples by linearly transforming them to have unit covariance; next, smooth the whitened samples using a radially symmetric kernel; and finally, transform them back. Effectively, this is equivalent to computing:

$$\hat{g}(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}\left(\boldsymbol{x}; \mathcal{D}^{(i)}, \beta\boldsymbol{S}\right), \tag{3}$$

where $\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ represents a Gaussian function with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. In the case of (3), the covariance of each component $\boldsymbol{S}$ is the same and is the unbiased sample/ensemble covariance:

$$\boldsymbol{S} = \frac{1}{n-1} \sum_{i=1}^{n} (\bar{\mathcal{D}} - \mathcal{D}^{(i)})(\bar{\mathcal{D}} - \mathcal{D}^{(i)})^{\top}, \tag{4}$$

where

$$\bar{\mathcal{D}} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{D}^{(i)}.$$

But what exactly is β? A number of methods for computing it have been developed, but most notably in [19, Ch. 4, pp. 76–88] which shows that the optimal[2] bandwidth $\beta_{\text{opt.}}$ for the smoothing of Gaussian distributed data with unit variance is:

$$\beta_{\text{opt.}} = \left(\frac{4}{n\,(d+2)}\right)^{2/(d+4)}. \tag{5}$$

This optimal bandwidth selection is referred to in the literature as Silverman's Rule of Thumb.

Silverman's Rule of Thumb is relatively simple to compute, which reduces the computational cost compared to alternative bandwidth selection algorithms. But even though (5) can be directly substituted into (3), thereby satisfying the kernel density estimator, it should still be used with caution. For example, for non-Gaussian distributions, (5) is no longer optimal, and instead is known to be too large, producing conservative estimates [3], [22]. While conservative estimates may be less accurate, they are often desirable because erring on the side of caution helps maintain filter stability and prevent divergence.

Nonetheless, because of its suboptimality, Silverman's Rule of Thumb serves only as an approximation, and as pointed out by [1], any chosen bandwidth may require an additional heuristic scaling α:

$$\beta \leftarrow \alpha\,\beta.$$

B. EnGMF

At time k, the EnGMF approximates the prior density $p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k-1})$ with a Dirac mixture:

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) \approx \frac{1}{n} \sum_{i=1}^{n} \delta(\boldsymbol{x}_k - \boldsymbol{\mathcal{X}}_k^{(i)}), \tag{6}$$

where $\delta(\cdot)$ is the Dirac delta distribution and $\{\boldsymbol{\mathcal{X}}_k^{(i)} \in \mathbb{R}^{n_{\boldsymbol{x}}} \colon i = 1, \ldots, n\}$ is an ensemble of n i.i.d. samples (particles). The notation $\boldsymbol{y}_{1:k-1}$ denotes the time history of discrete measurements $\boldsymbol{y} \in \mathbb{R}^{n_{\boldsymbol{y}}}$ from 1 to $k-1$. The EnGMF transforms this Dirac mixture into a GMM using KDE:

$$(6) \underset{\text{KDE}}{\longrightarrow} \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(\boldsymbol{x}_k; \hat{\boldsymbol{x}}_{k|k-1}^{(i)}, \hat{\boldsymbol{P}}_{k|k-1}^{(i)}). \tag{7}$$

To achieve this transformation, many different KDE approaches can be used. The original approach, in [1], performs the transformation using (3), which centers each mean at the location of its corresponding particle $\hat{\boldsymbol{x}}_{k|k-1}^{(i)} \leftarrow \boldsymbol{\mathcal{X}}_k^{(i)}$ and assigns covariances to all be the same, first by computing the sample covariance using (4) and then scaling it by some bandwidth parameter β. Reference [3] computes the bandwidth parameter using Silverman's Rule of Thumb (5) and shows that the EnGMF can produce more accurate and consistent estimates relative to the UKF, EnKF, and a state-of-the-art Gaussian sum filter when applied to a sparse-data orbit determination problem in Low Earth Orbit (LEO). Alternatively, instead of computing a constant bandwidth for all components, [4] uses EM to estimate the bandwidth parameter locally, which helps reduce conservativeness in the update and better fits the prior GMM; however, at the expense of being slow due to EM.

It should be noted that other approaches similar to the EnGMF also exist, but they do not use KDE for the transformation and instead use clustering. References [5], [6] present Particle Gaussian Mixture (PGM) Filters, which use K-means to transform the entire prior set of particles directly into a GMM. According to [2], EnGMF and PGM Filters are similar in that they can be categorized as types of "Ensemble Gaussian-Mixture Filters" (EGMFs), who perform the transformation in one of two ways: either through (i) KDE or (ii) clustering

[1]Although other kernels are also possible like the Epanechnikov kernel, which can sometimes be more efficient [19], [20], this work focuses on the multivariate Gaussian kernel because of its simplicity and wider acceptable understanding.

[2]Optimal in the sense of minimizing the mean integrated square error (MISE).

algorithms such as EM or K-means. Therefore, PGM filters can be thought of as a type of ensemble Gaussian mixture filtering method and are particularly useful for estimating multimodal distributions. However, in this work we denote any *EnGMF* as a kernel-based one, *i.e.*, one that performs the transformation using some form of KDE; whereas, we denote *PGM* for those that perform clustering.

Nevertheless, once the prior GMM is established, the standard Gaussian Sum Filter (GSF) equations from [23], [24] are then used to construct the posterior GMM estimate:

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) \approx \sum_{i=1}^{n} w_{k|k}^{(i)} \mathcal{N}(\boldsymbol{x}_k; \hat{\boldsymbol{x}}_{k|k}^{(i)}, \hat{\boldsymbol{P}}_{k|k}^{(i)}),$$

where each mean and covariance, $\hat{\boldsymbol{x}}_{k|k}^{(i)}$ and $\hat{\boldsymbol{P}}_{k|k}^{(i)}$, are individually computed through a linear Kalman-style update, while the updated weights are proportional to the measurement marginal PDF:

$$w_{k|k}^{(i)} \propto w_{k|k-1}^{(i)} p^{(i)}(\boldsymbol{y}_k \mid \boldsymbol{y}_{1:k-1}) \quad \text{and} \quad \sum_{i=1}^{n} w_{k|k}^{(i)} = 1,$$

where the prior weights $w_{k|k-1}^{(i)}$ are assumed to be uniform, *i.e.*, $1/n$ and also sum to 1. For output (not affecting the filter), the conditional mean and covariance can be computed:

$$\bar{\boldsymbol{x}}_{k|k} = \sum_{i=1}^{n} w_{k|k}^{(i)} \hat{\boldsymbol{x}}_{k|k}^{(i)} \quad \text{and} \tag{8}$$

$$\bar{\boldsymbol{P}}_{k|k} = \sum_{i=1}^{n} w_{k|k}^{(i)} (\hat{\boldsymbol{P}}_{k|k}^{(i)} + \hat{\boldsymbol{x}}_{k|k}^{(i)} \hat{\boldsymbol{x}}_{k|k}^{(i)\top}) - \bar{\boldsymbol{x}}_{k|k} \bar{\boldsymbol{x}}_{k|k}^{\top}. \tag{9}$$

Next, to prevent degeneracy, a new ensemble of i.i.d. samples are obtained from the posterior GMM estimate. There are many ways to achieve this, see for example [25]–[27]. Finally, to continue the recursion, these new samples are predicted to the next time step using the modeled transition density $\boldsymbol{\mathcal{X}}_{k+1}^{(i)} \sim p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)$, which typically involves propagating the particles through the system's dynamics and adding process noise.

C. Clustering on the Joint Distribution

An interesting approach is proposed by [8], where instead of performing clustering on the prior, they perform clustering on the joint distribution $p(\boldsymbol{x}_k, \boldsymbol{y}_k \mid \boldsymbol{y}_{1:k-1})$ and then use the conditioning property of Gaussian mixtures to condition the random vector (r.v.) $\boldsymbol{x}_k$ on $\boldsymbol{y}_k$, producing a GMM estimate of the posterior density $p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k})$. Since the first and second moments are approximated statistically, the conditioning skips the need to linearize the measurement model and individually compute Jacobians for each component. This approach they refer to as the "Ensemble Gaussian-Mixture Filter". But to avoid confusion with other methods in this work, we will interpret it as a type of PGM filter that performs clustering on the joint instead of the prior; we therefore denote it as the *Joint PGM Filter* or *JPGM* for short.

In closer detail, the JPGM starts by augmenting the set of prior particles with the measurement information to approximate the joint as a Dirac mixture:

$$p(\boldsymbol{x}_k, \boldsymbol{y}_k \mid \boldsymbol{y}_{1:k-1}) \approx \frac{1}{n} \sum_{i=1}^{n} \delta \left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{y}_k \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mathcal{X}}_k^{(i)} \\ h(\boldsymbol{\mathcal{X}}_k^{(i)}, \boldsymbol{\upsilon}_k^{(i)}) \end{bmatrix} \right), \tag{10}$$

where $h(\cdot,\cdot) : \mathbb{R}^{n_{\boldsymbol{x}}} \times \mathbb{R}^{n_{\boldsymbol{y}}} \to \mathbb{R}^{n_{\boldsymbol{y}}}$ is the known nonlinear mapping function and $\boldsymbol{\upsilon}_k^{(i)} \in \mathbb{R}^{n_{\boldsymbol{y}}}$ is noise.

Next, (10) is transformed into a GMM via a clustering algorithm like EM or K-means. This process identifies like-particles and uses their ensemble statistics to approximate the first and second moments of a Gaussian distribution with $K << n$ components:

$$(10) \underset{\text{EM}}{\longrightarrow} \sum_{i=1}^{K} w_k^{(i)} \mathcal{N} \left(\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{y}_k \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu}_{\boldsymbol{x}}^{(i)} \\ \boldsymbol{\mu}_{\boldsymbol{y}}^{(i)} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{xx}}^{(i)} & \boldsymbol{\Sigma}_{\boldsymbol{xy}}^{(i)} \\ \boldsymbol{\Sigma}_{\boldsymbol{yx}}^{(i)} & \boldsymbol{\Sigma}_{\boldsymbol{yy}}^{(i)} \end{bmatrix} \right). \tag{11}$$

Once that is done, the JPGM estimates the posterior density by using the conditioning property of Gaussian mixtures (see the Appendix), effectively performing a Kalman-style linear update on each component:

$$p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k}) \approx \sum_{i=1}^{K} \tilde{w}_k^{(i)} \mathcal{N}(\boldsymbol{x}_k; \hat{\boldsymbol{x}}_{k|k}^{(i)}, \hat{\boldsymbol{P}}_{k|k}^{(i)}), \tag{12}$$

where

$$\hat{\boldsymbol{x}}_{k|k}^{(i)} = \boldsymbol{\mu}_{\boldsymbol{x}}^{(i)} + \boldsymbol{\Sigma}_{\boldsymbol{xy}}^{(i)} (\boldsymbol{\Sigma}_{\boldsymbol{yy}}^{(i)})^{-1} (\boldsymbol{y}_k - \boldsymbol{\mu}_{\boldsymbol{y}}^{(i)}),$$

$$\hat{\boldsymbol{P}}_{k|k}^{(i)} = \boldsymbol{\Sigma}_{\boldsymbol{xx}}^{(i)} - \boldsymbol{\Sigma}_{\boldsymbol{xy}}^{(i)} (\boldsymbol{\Sigma}_{\boldsymbol{yy}}^{(i)})^{-1} \boldsymbol{\Sigma}_{\boldsymbol{yx}}^{(i)}, \quad \text{and}$$

$$\tilde{w}_k^{(i)} \propto w_k^{(i)} \mathcal{N}(\boldsymbol{y}_k; \boldsymbol{\mu}_{\boldsymbol{y}}^{(i)}, \boldsymbol{\Sigma}_{\boldsymbol{yy}}^{(i)}).$$

Finally, particles are resampled from the GMM approximation of the posterior and then are predicted to the next time step through the modeled transition density $\boldsymbol{\mathcal{X}}_{k+1}^{(i)} \sim p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)$.

One limitation, however, is that the clustering algorithms themselves, like EM and K-means, are costly computations and only get more costly as the number of particles increases. For example, in each iteration of the K-means algorithm, it experiences an average time complexity of $\mathcal{O}(n \times d \times K)$. For EM, its E step alone is $\mathcal{O}(n \times d \times K)$ and can have an M step that is much worse. These clustering algorithms are also not robust to *all* problems, requiring good initializations and heuristic tuning to get working. [12], [13]

Another limitation is that the augmented particles can cause ill-conditioning if the measurement noise is much smaller than the state uncertainty. This can typically happen when sensors are extremely accurate while the dynamics are strongly nonlinear, *e.g.*, in the cislunar regime. Ill-conditioning can also occur when either K is too large or n is too small, causing the clustering algorithm to not have enough particles to form each component. Alternatively, if K is set to be too small, then the filter may not effectively estimate the non-Gaussian posterior density with enough components (a common limitation in PGM filters), potentially ruining any benefits to using this approach.

Also, the JPGM requires the innovation matrices be individually inverted $(\boldsymbol{\Sigma}_{\boldsymbol{yy}}^{(i)})^{-1}$, which can slow down the filter whenever the number of components is high. Though, this can be avoided by using a clustering algorithm that produces the same covariance for each mixture component [12, Ch. 9, pp. 423–455], providing the ability to compute the inversion once and saving computational resources.

III. Methodology

A. Kernel-Based SEnGMF Algorithm

In this section, we propose a kernel-based SEnGMF that performs efficient KDE via Silverman's Rule of Thumb instead of clustering. The following steps outline the proposed algorithm.

0. Start with an initial ensemble of n i.i.d. particles:
$$p(\boldsymbol{x}_0) \approx \tfrac{1}{n}\textstyle\sum_{i=1}^{n} \delta(\boldsymbol{x}_0 - \boldsymbol{\mathcal{X}}_0^{(i)}).$$
1. Prediction step – sample $\boldsymbol{\mathcal{X}}_k^{(i)}$ from the transition density $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ by propagating particles through the known dynamics model and adding zero-mean Gaussian white noise:
$$\boldsymbol{\mathcal{X}}_k^{(i)} = f(\boldsymbol{\mathcal{X}}_{k-1}^{(i)}) + \boldsymbol{\omega}_k^{(i)}, \quad \boldsymbol{\omega}_k^{(i)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k).$$
2. Augmentation step – estimate the joint distribution as a Dirac mixture:
$$p(\boldsymbol{x}_k, \boldsymbol{y}_k \mid \boldsymbol{y}_{1:k-1}) \approx \frac{1}{n}\sum_{i=1}^{n} \delta\left(\begin{bmatrix}\boldsymbol{x}_k \\ \boldsymbol{y}_k\end{bmatrix} - \begin{bmatrix}\boldsymbol{\mathcal{X}}_k^{(i)} \\ h(\boldsymbol{\mathcal{X}}_k^{(i)})\end{bmatrix}\right). \quad (13)$$
3. Transformation step – convert the Dirac mixture into a GMM using (3):
$$(13) \underset{\text{Silv.}}{\longrightarrow} \frac{1}{n}\sum_{i=1}^{n} \mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}_k \\ \boldsymbol{y}_k\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mathcal{X}}_k^{(i)} \\ h(\boldsymbol{\mathcal{X}}_k^{(i)})\end{bmatrix}, \beta\begin{bmatrix}\boldsymbol{S}_{\boldsymbol{xx}} & \boldsymbol{S}_{\boldsymbol{xy}} \\ \boldsymbol{S}_{\boldsymbol{yx}} & \boldsymbol{S}_{\boldsymbol{yy}}\end{bmatrix}\right), \quad (14)$$
where $\boldsymbol{S}_{\cdot,\cdot}$ are unbiased sample covariances computed using (4) and β is Silverman's Rule of Thumb from (5) with $d = n_{\boldsymbol{x}}$.
4. Update step – estimate the posterior density by using the GMM conditioning property on the joint:
$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) \approx \sum_{i=1}^{n} w_{k|k}^{(i)} \mathcal{N}(\boldsymbol{x}_k; \hat{\boldsymbol{x}}_{k|k}^{(i)}, \hat{\boldsymbol{P}}_{k|k}^{(i)}), \quad (15)$$
where
$$\hat{\boldsymbol{x}}_{k|k}^{(i)} = \boldsymbol{\mathcal{X}}_k^{(i)} + \beta\boldsymbol{S}_{\boldsymbol{xy}}(\beta\boldsymbol{S}_{\boldsymbol{yy}} + \boldsymbol{R}_k)^{-1}(\boldsymbol{y}_k - h(\boldsymbol{\mathcal{X}}_k^{(i)})),$$
$$\hat{\boldsymbol{P}}_{k|k}^{(i)} = \beta\boldsymbol{S}_{\boldsymbol{xx}} - \beta\boldsymbol{S}_{\boldsymbol{xy}}(\boldsymbol{S}_{\boldsymbol{yy}})^{-1}\boldsymbol{S}_{\boldsymbol{yx}}, \quad \text{and}$$
$$w_{k|k}^{(i)} \propto \mathcal{N}(\boldsymbol{y}_k; h(\boldsymbol{\mathcal{X}}_k^{(i)}), \beta\boldsymbol{S}_{\boldsymbol{yy}} + \boldsymbol{R}_k).$$
5. Obtain new particles from the GMM posterior $\boldsymbol{\mathcal{X}}_k^{(i)} \sim p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ and output the conditional mean and covariance using (8) and (9), respectively.
6. Go back to 1.

B. Benefits and Properties

Just like the JPGM from Section II-C, this proposed kernel-based SEnGMF directly approximates the first and second moments of the joint distribution, rather than linearizing the measurement model through Jacobians. The SEnGMF, however, is more computationally efficient. First, it uses KDE and efficiently computes its bandwidth parameter β by using Silverman's Rule of Thumb, which we already established is less costly than alternative bandwidth selection algorithms. Second, if the number of components does not change, then the bandwidth can be *precomputed* offline when the filter initializes, thereby only needing to compute it once for all time. Third, since the KDE transformation is effectively a scalar multiplied by a sample covariance, the time complexity is $\mathcal{O}(n \times d^2)$, which is far superior than a clustering algorithm like EM or K-means that are at best $\mathcal{O}(n \times d \times K \times T)$, where T is the number of convergence iterations.

Not only is the kernel-based SEnGMF more efficient than the JPGM from Section II-C but also more so than the standard EnGMF from Section II-B. This is because each component shares the *same* covariance, meaning the Kalman gain, the inverse of the innovation matrix, and the posterior covariance only need to be computed once for the entire mixture.

Another property of this filter is that if $\beta = 0$, then the Kalman-style update would disappear and the kernel-based SEnGMF degenerates to a standard particle filter. If $\beta = 1$, then the Kalman gain inflates to the statistical EnKF gain from [15]. This is similar to how β bridges the standard EnGMF [1] between the particle filter and the linearized EnKF gains [15]. However, even though the bandwidth β seems like it would fully bridge the algorithms of the particle filter and EnKF, it does not exactly. This is because the EnKF does not resample particles and was later amended by [15]–[17] to add random perturbations to the measurements in the update, which are nonstandard features in the EnGMF.

C. Assumptions

One assumption made in the proposed algorithm is that the measurement noise is additive: $\boldsymbol{y}_k = h(\boldsymbol{x}_k) + \boldsymbol{\upsilon}_k$, where $\boldsymbol{\upsilon}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k)$. This assumption allows $\boldsymbol{R}_k$ to be added separately to the sample covariance $\boldsymbol{S}_{\boldsymbol{yy}}$ so that the innovation matrix becomes $\beta\boldsymbol{S}_{\boldsymbol{yy}} + \boldsymbol{R}_k$.

However, it is possible to relax this assumption. If the noise was *not* additive: $\boldsymbol{y}_k = h(\boldsymbol{x}_k, \boldsymbol{\upsilon}_k)$ like in Section II-C, then the innovation matrix would be $\beta\boldsymbol{S}_{\boldsymbol{yy}}$ since the augmented r.v. already contains randomness from the state and measurement elements. This would affect β, which would then have $d = n_{\boldsymbol{x}} + n_{\boldsymbol{y}}$, instead of $d = n_{\boldsymbol{x}}$. But this generalization is not explored further.

D. Limitations to Consider

While the proposed filter has many benefits, it has limitations that should still be addressed. The first is that computing a global Kalman gain may be worse than computing local ones. The kernel-based SEnGMF computes a global MC Kalman gain, which means it is shared across all components.

However, this forces all components to use the same linear approximation of the measurement model, which can under- or over-correct states that are far from the ensemble mean. Therefore, it might be better to use a standard EnGMF that computes local Kalman gains through Jacobians, since these gains account for the local linearization effects of the measurement model across the distribution.

Another limitation is that, in lower sample sizes, ill-conditioning can occur, which causes problems in the update and ruins filter consistency, ultimately leading to filter divergence. Therefore, if there are not enough samples, it might again be better to use a standard EnGMF that computes local Kalman gains through Jacobian matrices, which can help keep the filter numerically stable. But note that this limitation is not unique to the SEnGMF and also appears in other MC-based filters like the EnKF [15], [18]; however, it can typically be resolved by simply adding more particles.

Finally, another limitation can occur when the prior distribution is multimodal. Using the entire ensemble's sample covariance can lead to undesired stretching if the modes are spatially separated significantly. This is also a limitation of the kernel-based EnGMF. Instead, it might be better to use clustering, albeit at the expense of clustering, as is done in the PGM filters [5], [6], where they handle multimodal priors through K-means.

IV. Results

The following tests the proposed kernel-based SEnGMF in a cislunar relative navigation problem. It is tested against the Extended Kalman Filter (EKF) [28, Ch. 6, pp. 182–191], the Unscented Kalman Filter (UKF) [29], [30], the EnGMF [1], [3], the Bootstrap particle filter [31], and the JPGM filter from [8] that performs clustering via EM.

The problem involves a Chaser space vehicle in a Near Rectilinear Halo Orbit (NRHO), tasked with estimating its relative position and velocity to a Target in the same orbit. We assume the Chaser vehicle knows perfectly its own inertial position and velocity inside the Earth-Moon Circular Restricted 3 Body Problem (CR3BP) rotating frame. The Chaser only uses a vision-based sensor (e.g., a camera) that collects noisy angle and angle rate measurements when the Target passes through its field of view (FOV). An illustration of this problem is in Fig. 1.

This is a challenging problem for a couple reasons. First, cislunar NRHOs are known for their chaotic behavior and sensitivity to initial conditions, where slight deviations in the initial conditions lead to large deviations in the estimated states [32], [33]. Second, the angles-only measurements are not just nonlinear but are also sparse in time and leave the range and range rate directions weakly observable. Therefore, this combination of strongly nonlinear dynamics and measurements cause non-Gaussian probability distributions, necessitating the use of nonlinear filters where a traditional linear filter will fail.

A. Dynamics Model Parameters

Both space objects' inertial states are described by the 3D Cartesian position-velocity state:

$$\boldsymbol{x} = [\boldsymbol{r}^\top, \boldsymbol{v}^\top]^\top \in \mathbb{R}^6,$$

where $\boldsymbol{r} = [x, y, z]^\top \in \mathbb{R}^3$ and $\boldsymbol{v} = [\dot{x}, \dot{y}, \dot{z}]^\top \in \mathbb{R}^3$ represent their scaled Cartesian positions and velocities with respect to the Earth-Moon barycenter in the CR3BP rotating frame. They are scaled for numerical stability, where any and all distances and times are normalized by length and time units: $\text{LU} = 384\,400 \times 10^3\,\text{m}$ and $\text{TU} = (\text{LU}^3/(\mu_\oplus + \mu_\leftmoon))^{1/2}\,\text{s}$, respectively.

The continuous-time CR3BP dynamics are numerically integrated with an embedded Runge-Kutta 8(7) method [34], and are modeled:

$$\dot{\boldsymbol{r}} = \boldsymbol{v},$$

$$\dot{\boldsymbol{v}} = \begin{bmatrix} x + 2\dot{y} \\ y - 2\dot{x} \\ 0 \end{bmatrix} - (1-\mu)\frac{\boldsymbol{r} + [\mu, 0, 0]^\top}{r_\oplus^3} - \mu\frac{\boldsymbol{r} - [1-\mu, 0, 0]^\top}{r_\leftmoon^3},$$

where $r_\oplus$ and $r_\leftmoon$ are the distances of each object to the Earth and the Moon: $r_\oplus = ((x+\mu)^2 + y^2 + z^2)^{1/2}$ and $r_\leftmoon = ((x-1+\mu)^2 + y^2 + z^2)^{1/2}$. The scaling constant μ is the Moon geocentric gravitational parameter and is computed: $\mu = \mu_\leftmoon/(\mu_\oplus + \mu_\leftmoon)$, where both the Earth and the Moon have gravitational parameters $\mu_\oplus = G\,m_\oplus$ and $\mu_\leftmoon = G\,m_\leftmoon$, respectively. The universal gravitational constant $G = 6.6743 \times 10^{-11}\,\text{m}^3\,\text{s}^{-2}\,\text{kg}^{-1}$ and the mass of the Earth and Moon are $m_\oplus = 5.972 \times 10^{24}\,\text{kg}$ and $m_\leftmoon = 7.342 \times 10^{22}\,\text{kg}$, respectively.

The initial true state of the Chaser is placed at apolune:

$$\boldsymbol{x}_{0,\text{Chas.}} = [1.0110350588, 0, -0.17315, \\ 0, -0.0780141199, 0]^\top;$$

whereas, the initial true state of the Target is propagated ahead 80% of the Chaser's orbital period $\text{T} = 1.3632096570$:

$$\boldsymbol{x}_{0,\text{Targ.}} = [1.0072493730, 0.0199838575, -0.1522805465, \\ 0.0278765692, -0.0630216716, -0.1567468341]^\top.$$

Neither the Chaser's nor Target's truth experience any process noise along their trajectories.

In each Monte Carlo simulation, every filter starts with an initial Gaussian distribution centered at the following dimensionless relative coordinates $\hat{\boldsymbol{x}}_{0|0}$, with covariance $\hat{\boldsymbol{P}}_{0|0}$:

$$\begin{aligned} \hat{\boldsymbol{x}}_{0|0} &= \boldsymbol{x}_{0,\text{Targ.}} - \boldsymbol{x}_{0,\text{Chas.}} \\ &= [-0.0037856858,\ 0.0199838575,\ 0.0208694535, \\ &\quad 0.0278765692,\ 0.0149924483,\ -0.1567468341]^\top, \\ \hat{\boldsymbol{P}}_{0|0} &= \text{diag}([1\times10^{-4},\ 1\times10^{-4},\ 1\times10^{-4}, \\ &\quad 1\times10^{-6},\ 1\times10^{-6},\ 1\times10^{-6}]^2). \end{aligned}$$

The different filters also do not assume any additive discrete process noise: $\boldsymbol{Q}_k = \boldsymbol{0}_{6\times6}$. Each filter operates at a fixed rate of $\text{dt} = 100\ \text{min}$ and the simulation lasts for 5 orbits.

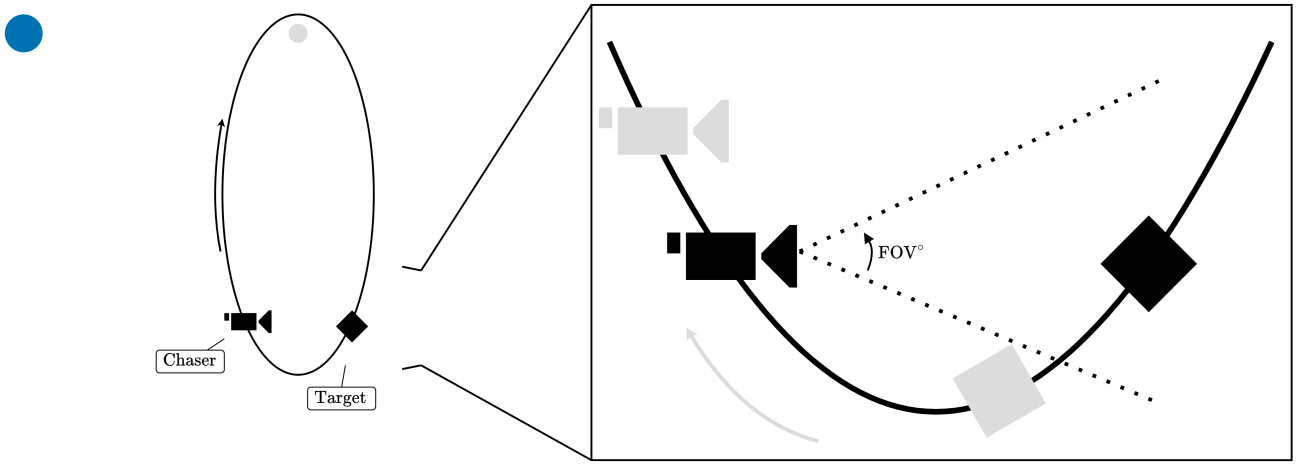


Fig. 1: Illustration of the problem scenario. The Target comes in and out of the Chaser's FOV as they orbit the Moon.

B. Measurement Model Parameters

Measurements are sparse in time, which means they are infrequent over long periods. The camera on the Chaser is pointed fixed in the positive Barycenter-inertial Y-direction and, as the Target crosses its FOV of 90°, measurements are recorded at a rate of $\mathrm{dt} = 100$ min.

The measurement vector is $\boldsymbol{y} = [\,\alpha, \varepsilon, \dot{\alpha}, \dot{\varepsilon}\,]^\top$, which contains both azimuth α and elevation ε angles of the Target mapped to the Chaser as well as their respective rates $\dot{\alpha}$ and $\dot{\varepsilon}$:

$$\alpha = \tan^{-1}\left(\frac{y}{x}\right), \quad \varepsilon = \sin^{-1}\left(\frac{z}{\rho}\right), \quad \dot{\alpha} = \frac{x\dot{y} - y\dot{x}}{x^2 + y^2},$$
$$\text{and} \quad \dot{\varepsilon} = \frac{\dot{z}(x^2 + y^2) - z(x\dot{x} + y\dot{y})}{\rho^2\sqrt{x^2 + y^2}},$$

where (abusing notation) $\boldsymbol{r}_{\text{Targ.}} - \boldsymbol{r}_{\text{Chas.}} = [\,x, y, z\,]^\top$, $\boldsymbol{v}_{\text{Targ.}} - \boldsymbol{v}_{\text{Chas.}} = [\,\dot{x}, \dot{y}, \dot{z}\,]^\top$, and $\rho = \|\boldsymbol{r}_{\text{Targ.}} - \boldsymbol{r}_{\text{Chas.}}\|$. The measurements are further corrupted by additive zero-mean Gaussian white noise sequences with standard deviations of 0.9 arcseconds for the angles and $0.01\ \mathrm{rad\,s^{-1}}$ for their rates. Note that we do not consider any measurement biases or time delays due to light travel. Also note that in this problem we avoid any detectability issues by assuming the Target has sufficient size and reflectivity, but for more information on the detectability of cislunar space objects, we direct the reader to see for example [35].

C. Filter Parameters

The EKF approximates its state transition matrix using the matrix exponential of the dynamics Jacobian times the filter rate $\mathrm{dt} = 100$ min, which provides a reasonable approximation for small step sizes. The UKF is the 3-parameter formulation from [30], which uses $\alpha' = 1$, $\beta' = 2$, and $\kappa' = 3 - n_{\boldsymbol{x}}$, where $n_{\boldsymbol{x}} = 6$, and re-Gaussianizes after every prediction step. The bootstrap particle filter uses $n = 1000$ particles and performs multinominal resampling [36] after the update, but only when the effective sample size is less than $n/2$. The EnGMF uses $n = 200$ components and computes its bandwidth using Silverman's Rule of Thumb. The JPGM also has $n = 200$ components and performs EM on the predicted components using MATLAB's `fitgmdist` function, with optional inputs: $K = 2$ clusters, `'Start'='randSample'`, `'Replicates'=5`, `'RegularizationValue'=1e-10`. These inputs were heuristically chosen to give us stable results. Finally, the proposed kernel-based SEnGMF uses $n = 200$ components and computes its bandwidth with Silverman's Rule of Thumb. The EnGMF, JPGM, and SEnGMF each resample after the update step.

D. Performance Criteria

The goal is to have a filter that is not only accurate but also consistent and computationally efficient. Therefore, in this problem, filter accuracy is evaluated using the Root Mean Square Error (RMSE) of the estimated conditional mean $\bar{\boldsymbol{x}}_{k|k}$ with respect to the truth $\boldsymbol{x}_k$, and is computed:

$$\mathit{RMSE}_k = \left(\frac{1}{n_{\boldsymbol{x}}}\sum_{i=1}^{n_{\boldsymbol{x}}}\left(\boldsymbol{x}_k(i) - \bar{\boldsymbol{x}}_{k|k}(i)\right)^2\right)^{1/2},$$

where a lower RMSE means the estimate more closely aligns with the truth, indicating better accuracy.

Filter consistency is evaluated using the Scaled Normalized (state) Estimation Error Squared (SNEES):

$$\mathit{SNEES}_k = \frac{1}{n_{\boldsymbol{x}}}(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_{k|k})^\top \bar{\boldsymbol{P}}_{k|k}^{-1}(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_{k|k}),$$

where the SNEES is an $n_{\boldsymbol{x}}$-scaled version of the NEES from [37, Ch. 3, pp. 165–166]. The optimal filter consistency for a linear Gaussian system is 1; therefore, a filter is deemed *conservative* if its SNEES is less than 1 and *confident* if it is greater than 1. While our presented cislunar problem is not linear Gaussian, the SNEES remains a valid measure of relative consistency between filters, showing us how consistent one filtering method is compared to another.

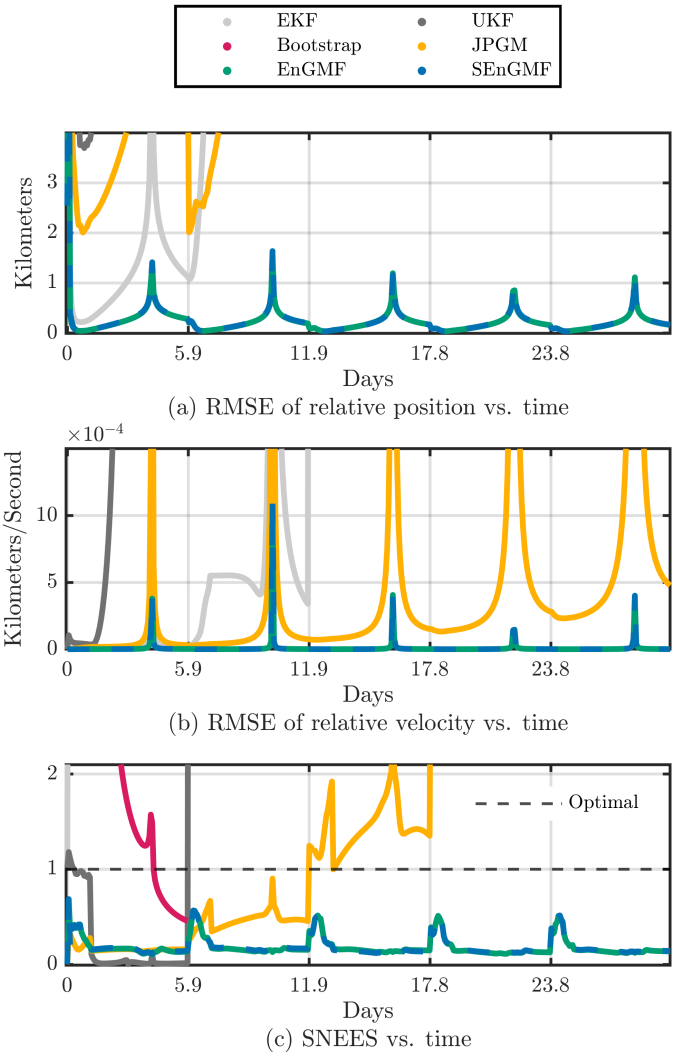
Filter computational efficiency is evaluated using MATLAB's `tic` and `toc` functions, which measure the wall-clock time. In this problem, the wall-clock time is the total elapsed time it took for each Monte Carlo simulation to run from start to finish. It includes active computation on the CPU and any waiting periods, making it a good measure of real-world performance. The following results are averaged over 1,000 Monte Carlo simulations and were obtained using an Intel Core i7-9700 CPU at a base operating rate of 3.0 GHz with 16 GB of RAM.

E. Numerical Results

It is apparent in Fig. 2 that each the EKF, UKF, Bootstrap and JPGM filters diverge over the 5 orbits. This occurs for both the EKF and UKF because they estimate only a single Gaussian density, which is insufficient for this strongly nonlinear and non-Gaussian problem. The Bootstrap particle filter fails because there is only a small number of particles existing where the likelihood is large, resulting the curse of dimensionality. This effectively occurs when the dynamics have no process noise and the measurements are accurate, creating steep peaks in the likelihood. For more on this, we refer the reader to [38], [39]. The JPGM likely fails because the measurement noise is smaller than the state uncertainty, causing ill-conditioning during the EM clustering step. Another reason is likely because K is tuned to be too small, causing the filter to inaccurately estimate the posterior density[3]. Regardless, the JPGM demonstrates it is not a practical solution to this problem, having diverging estimates and slow performance, with an average wall-clock time of around $5.28\,\mathrm{s}$.

Fig. 2 also shows the results of the EnGMF and the proposed SEnGMF. These two filters experience similar performance, outperforming the other filters. They both accurately and consistently produce relative estimates, and incur the same peaks and valleys. The peaks and valleys are due to the Target not always being visible, causing large uncertainties when out of the camera's FOV and accurate estimates when visible. Overall, the EnGMF has slightly better accuracy, producing time-averaged position and velocity RMSEs of $0.246\,\mathrm{km}$ and $9.63 \times 10^{-6}\,\mathrm{km\,s^{-1}}$, respectively, compared to the SEnGMF's $0.251\,\mathrm{km}$ and $9.87 \times 10^{-6}\,\mathrm{km\,s^{-1}}$, which are about 1.99% worse in position and 2.43% worse in velocity. This is likely due to the SEnGMF computing a global Kalman gain rather than local ones. Although it is slightly less accurate, the SEnGMF is a bit more consistent, having a time-averaged SNEES 0.184 compared to the EnGMF's 0.182. But regardless, these statistics are marginally close; whereas, the real benefit is that the SEnGMF is 11.6% more computationally efficient, with an average simulation wall-clock-time of $3.35\,\mathrm{s}$ compared to the EnGMF's $3.79\,\mathrm{s}$.

[3]It is typical for EM to have $K << n$ [12], where we found setting $K = 2$ was necessary for it to work properly and converge.



(a) RMSE of relative position vs. time

(b) RMSE of relative velocity vs. time

(c) SNEES vs. time

Fig. 2: Filter accuracy (RMSE) and consistency (SNEES) vs. time.

V. Conclusion

In this work we propose a kernel-based Statistical Ensemble Gaussian Mixture Filter (SEnGMF) that transforms the joint distribution into a Gaussian mixture model (GMM) using an efficient kernel density estimation (KDE) technique. By its construction, the kernel-based SEnGMF is computationally less complex than ensemble Gaussian mixture methods using clustering and the standard kernel-based EnGMF. This is because it avoids any clustering of particles, linearizing a measurement model, computing Jacobians, and inverting individual innovation matrices. From the results, we show the SEnGMF produces roughly the same accuracy and consistency as the EnGMF, all while being more computationally efficient.

Future work can explore how the proposed filter might be extended to a broader class of nonlinear problems, particularly those with stronger higher-order measurement nonlinearities or those with bimodal and multimodal state distributions.

APPENDIX

Conditioning property of GMMs. Consider a scenario where the joint distribution of two r.v.'s is exactly a Gaussian mixture:

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{i=1}^{n} w^{(i)} \mathcal{N}(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{\mu}^{(i)}, \boldsymbol{\Sigma}^{(i)}). \tag{16}$$

Alternatively, the joint can be equivalently rewritten as the following by augmenting the r.v.'s:

$$(16) = \sum_{i=1}^{n} w^{(i)} \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu}_1^{(i)} \\ \boldsymbol{\mu}_2^{(i)} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11}^{(i)} & \boldsymbol{\Sigma}_{12}^{(i)} \\ \boldsymbol{\Sigma}_{21}^{(i)} & \boldsymbol{\Sigma}_{22}^{(i)} \end{bmatrix} \right). \tag{17}$$

From the above GMM, the conditioning property of Gaussian mixtures can be used to condition the r.v. $\boldsymbol{x}_1$ on $\boldsymbol{x}_2$:

$$p(\boldsymbol{x}_1 \mid \boldsymbol{x}_2) = \sum_{i=1}^{n} \tilde{w}^{(i)} \mathcal{N}(\boldsymbol{x}_1; \tilde{\boldsymbol{\mu}}_1^{(i)}, \tilde{\boldsymbol{\Sigma}}_{11}^{(i)}), \tag{18}$$

where

$$\tilde{\boldsymbol{\mu}}_1^{(i)} = \boldsymbol{\mu}_1^{(i)} + \boldsymbol{\Sigma}_{12}^{(i)} (\boldsymbol{\Sigma}_{22}^{(i)})^{-1} (\boldsymbol{x}_2 - \boldsymbol{\mu}_2^{(i)}),$$

$$\tilde{\boldsymbol{\Sigma}}_{11}^{(i)} = \boldsymbol{\Sigma}_{11}^{(i)} - \boldsymbol{\Sigma}_{12}^{(i)} (\boldsymbol{\Sigma}_{22}^{(i)})^{-1} \boldsymbol{\Sigma}_{21}^{(i)}, \quad \text{and}$$

$$\tilde{w}^{(i)} = \frac{w^{(i)} \mathcal{N}(\boldsymbol{x}_2; \boldsymbol{\mu}_2^{(i)}, \boldsymbol{\Sigma}_{22}^{(i)})}{\sum_{j=1}^{n} w^{(j)} \mathcal{N}(\boldsymbol{x}_2; \boldsymbol{\mu}_2^{(j)}, \boldsymbol{\Sigma}_{22}^{(j)})}.$$

REFERENCES

[1] J. L. Anderson and S. L. Anderson, "A Monte Carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts," *Monthly Weather Review*, vol. 127, no. 12, pp. 2744–2745, 1999.

[2] S. Reich, "A Gaussian-mixture ensemble transform filter," *Quarterly Journal of the Royal Meteorological Society*, vol. 138, no. 662, pp. 222–233.

[3] S. Yun, R. Zanetti, and B. A. Jones, "Kernel-based ensemble Gaussian mixture filtering for orbit determination with sparse data," *Advances in Space Research*, vol. 69, no. 12, pp. 4179–4197, 2022.

[4] A. A. Popov and R. Zanetti, "An adaptive covariance parameterization technique for the ensemble Gaussian mixture filter," *SIAM Journal on Scientific Computing*, vol. 46, no. 3, pp. A1949–A1971, 2024.

[5] D. Raihan and S. Chakravorty, "Particle Gaussian mixture filters-i," *Automatica*, vol. 98, pp. 331–340, 2018.

[6] ——, "Particle Gaussian mixture filters-ii," *Automatica*, vol. 98, pp. 341–349, 2018.

[7] S. Yun and R. Zanetti, "Nonlinear filtering of light-curve data," *Advances in Space Research*, vol. 66, no. 7, pp. 1672–1688, 2020.

[8] A. Olivier and A. W. Smyth, "Review of nonlinear filtering for SHM with an exploration of novel higher-order Kalman filtering algorithms for uncertainty quantification," *Journal of Engineering Mechanics*, vol. 143, no. 11, p. 04017128, 2017.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[10] S. K. Ng, T. Krishnan, and G. J. McLachlan, *The EM Algorithm*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 139–172.

[11] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[12] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*. Springer, 2006, vol. 4, no. 4.

[13] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.

[14] D. Durant, F. Giraldo-Grueso, and R. Zanetti, "An adaptive and deterministic ensemble Gaussian mixture filter for cislunar relative navigation," in *2025 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2025, pp. 1–8.

[15] G. Evensen, "Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics," *Journal of Geophysical Research: Oceans*, vol. 99, no. C5, pp. 10 143–10 162, 1994.

[16] G. Burgers, P. J. Van Leeuwen, and G. Evensen, "Analysis scheme in the ensemble Kalman filter," *Monthly Weather Review*, vol. 126, no. 6, pp. 1719–1724, 1998.

[17] P. Houtekamer and H. L. Mitchell, "Data assimilation using an ensemble Kalman filter technique," *Monthly Weather Review*, vol. 126, no. 3, pp. 796–811, 1998.

[18] M. Asch, M. Bocquet, and M. Nodet, *Data Assimilation: Methods, Algorithms, and Applications*. SIAM, 2016.

[19] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Routledge, 1986.

[20] A. A. Popov and R. Zanetti, "Are non-Gaussian kernels suitable for ensemble mixture model filtering?" in *2024 27th International Conference on Information Fusion (FUSION)*. IEEE, 2024, pp. 1–8.

[21] K. Fukunaga, *Instruction to Statistical Pattern Recognition*. Elsevier, 1972.

[22] P. Janssen, J. S. Marron, N. Veraverbeke, and W. Sarle, "Scale measures for bandwidth selection," *Journal of Nonparametric Statistics*, vol. 5, no. 4, pp. 359–380, 1995.

[23] H. W. Sorenson and D. L. Alspach, "Recursive Bayesian estimation using Gaussian sums," *Automatica*, vol. 7, no. 4, pp. 465–479, 1971.

[24] D. Alspach and H. Sorenson, "Nonlinear Bayesian estimation using Gaussian sum approximations," *IEEE Transactions on Automatic Control*, vol. 17, no. 4, pp. 439–448, 1972.

[25] I. Gilitschenski, J. Steinbring, U. D. Hanebeck, and M. Simandl, "Deterministic Dirac mixture approximation of Gaussian mixtures," in *2014 17th International Conference on Information Fusion (FUSION)*, 2014, pp. 1–7.

[26] B. Liu, B. Ait-El-Fquih, and I. Hoteit, "Efficient kernel-based ensemble Gaussian mixture filtering," *Monthly Weather Review*, vol. 144, no. 2, 2016.

[27] S. Yun and R. Zanetti, "Sequential Monte Carlo filtering with Gaussian mixture sampling," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 9, pp. 2069–2077, 2019.

[28] A. Gelb, *Applied Optimal Estimation*. MIT Press, 1974.

[29] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI*, vol. 3068. Spie, 1997, pp. 182–193.

[30] R. Van der Merwe, "Sigma-point Kalman filters for probabilistic inference in dynamic state-space models," Ph.D. dissertation, The OGI School of Science and Engineering at Oregon Health and Science University, 2004.

[31] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings F (Radar and Signal Processing)*, vol. 140, pp. 107–113, 1993.

[32] A. F. Haapala and K. C. Howell, "A framework for constructing transfers linking periodic libration point orbits in the spatial circular restricted three-body problem," *International Journal of Bifurcation and Chaos*, vol. 26, no. 05, p. 1630013, 2016.

[33] E. M. Z. Spreen, "Trajectory design and targeting for applications to the exploration program in cislunar space," Ph.D. dissertation, Purdue University, 2021.

[34] J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[35] C. Frueh, K. Howell, K. J. DeMars, and S. Bhadauria, "Cislunar space situational awareness," in *31st AIAA/AAS Space Flight Mechanics Meeting*, 2021, pp. 21–290.

[36] J. D. Hol, T. B. Schon, and F. Gustafsson, "On resampling algorithms for particle filters," in *2006 IEEE Nonlinear Statistical Signal Processing Workshop*. IEEE, 2006, pp. 79–82.

[37] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. John Wiley & Sons, 2001.

[38] C. Snyder, T. Bengtsson, P. Bickel, and J. Anderson, "Obstacles to high-dimensional particle filtering," *Monthly Weather Review*, vol. 136, no. 12, 2008.

[39] P. J. Van Leeuwen, "Particle filtering in geophysical systems," *Monthly Weather Review*, vol. 137, no. 12, pp. 4089–4114, 2009.