

Nonlinear Gaussian Filtering via Radial Basis Function Approximation

Huazhen Fang, Jia Wang and Raymond A. de Callafon

Abstract—This paper presents a novel type of Gaussian filter — the radial basis Gaussian filter (RB-GF) — for nonlinear state estimation. In the RB-GF, we propose to use radial basis functions (RBFs) to approximate the nonlinear process and measurement functions of a system, considering the superior approximation performance of RBFs. Optimal determination of the approximators is achieved by RBF neural network (RBFNN) learning. Using the RBF based function approximation, the challenging problem of integral evaluation in Gaussian filtering can be well solved, guaranteeing the filtering performance of the RB-GF. The proposed filter is studied through numerical simulation, in which a comparison with other existing methods validates its effectiveness.

I. INTRODUCTION

Filter design for nonlinear state estimation has been a long-standing challenge in many fields including control systems, signal processing, navigation and guidance, etc., with a large amount of research effort having been dedicated to this topic during the past decades [1]. The extended Kalman filter (EKF) is arguably the most popular technique [2]. However, the performance of the EKF is often unsatisfactory in terms of convergence speed and robustness to serious nonlinearities. A number of other KF variants have thus been proposed for improvements, e.g., unscented KF (UKF) and ensemble KF (EnKF).

Much attention in recent years has been directed towards *Gaussian filtering* for the purpose of obtaining analytic or closed-form nonlinear filters. Gaussian filters build on Bayesian state estimation and assumed density filtering (ADF). A Bayesian estimator sequentially updates the conditional probability density functions (pdf's) of unknown state variables given the output measurements [3]. The ADF assumes a particular form of density, which are often mathematically tractable to deal with, for the pdf's involved in the Bayesian estimator, and then computes the state estimates [4]. If the assumed densities are Gaussian, the ADF will lead to the Gaussian filters.

A crucial problem in Gaussian filtering is to evaluate a number of integrals. The integrand of each integral is the multiplication of a nonlinear function (stemming from the nonlinear system equations) and a Gaussian function. To address the problem, two main approaches have been proposed. One of them is based on Monte Carlo sampling that

approximates pdf's by a set of random samples. In filtering, a set of state estimates (i.e., samples) are generated in light of the *a priori* conditional pdf. The KF computation is implemented to each of them, and then the individual estimation results will be aggregated to yield the final state estimate. The aforementioned integrals are implicitly approximated during the process. Two typical methods that fall into this category are the EnKF [5] and the UKF [6], the difference between which is that the latter uses deterministic sampling. The other approach is direct numerical integration. Making use of the Gauss-Hermite quadrature rule, the Gauss-Hermite filter (GHF) is capable of giving almost accurate evaluation of the integrals arising in Gaussian filtering through a weighted summation of the nonlinear function evaluated at some fixed points [7; 8]. In [9], the cubature KF (CKF) is proposed, which adopts a spherical-radial cubature rule for numerical computation of these integrals.

The objective of this paper is to develop a new Gaussian filter that is realized by a radial basis function (RBF) based approach. Widely used in pattern classification and curve fitting problems [10], RBFs also find important applications in the development of neural network based control systems, e.g., [11; 12; 13; 14]. We propose that integral evaluation in Gaussian filtering can be reduced to function approximation, the construction of which can be achieved using a set of RBFs. In this paper, we will use the Gaussian RBF (GRBF), because Gaussian-type functions, which frequently appears in Gaussian filtering, are easy to manipulate to derive integration in closed form. With an equivalent structure to a RBF Neural Network (RBFNN), the RBF based function approximator can be established through training the RBFNN. The obtained filter, which we refer to as the *radial basis Gaussian filter* (RB-GF), has high estimation performance and satisfactory computational efficiency.

Essentially a Gaussian filter with RBFNNs employed to assist in dealing with integral evaluation, the proposed RB-GF differs significantly from existing RBFNN based nonlinear filtering schemes, e.g., [15], in which RBFNNs are used to model unknown or uncertain system dynamics. In addition, instead of doing fixed-point quadrature or cubature approximation like in [8; 9], it is a customized filter incorporating construction of approximators for different nonlinear functions in different systems.

The remainder of the paper is as follows. Section II presents the Gaussian filtering technique, the derivation of which from Bayesian estimation theory and ADF is introduced. We then develop the RB-GF in Section III, showing the novel application of the RBFs to Gaussian filtering. A simulation study is illustrated in Section IV to show

H. Fang is with the Department of Mechanical & Aerospace Engineering, University of California, San Diego, CA 92093, USA.

J. Wang is with the School of Control Science & Engineering, Dalian University of Technology, Dalian, China, and is currently visiting the Department of Mechanical & Aerospace Engineering, University of California, San Diego, CA 92093, USA.

R.A. de Callafon is with the Department of Mechanical & Aerospace Engineering, University of California, San Diego, CA 92093, USA. callafon@ucsd.edu

the effectiveness of the RB-GF. Finally, some concluding remarks are offered in Section V.

II. NONLINEAR GAUSSIAN FILTERING

Let us consider the following nonlinear discrete-time system:

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) + \mathbf{w}_k, \\ \mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k, \end{cases} \quad (1)$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$ is the unknown system state, and $\mathbf{y}_k \in \mathbb{R}^{n_y}$ is the output. The process noise \mathbf{w}_k and the measurement noise \mathbf{v}_k are mutually independent, zero-mean white Gaussian sequences with covariances \mathbf{Q}_k and \mathbf{R}_k , respectively. For a Gaussian random vector $\mathbf{x} \in \mathbb{R}^n$, we use the notation $\mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A}) := (2\pi)^{-\frac{n}{2}} |\mathbf{A}|^{-\frac{1}{2}} \cdot \exp[-\frac{1}{2}(\mathbf{x} - \mathbf{a})^T \mathbf{A}^{-1}(\mathbf{x} - \mathbf{a})]$, where $\mathbf{x}, \mathbf{a} \in \mathbb{R}^n$ and $|\mathbf{A}|$ denotes the determinant of $\mathbf{A} \in \mathbb{R}^{n \times n}$. Then we have $p(\mathbf{w}_k) = \mathcal{N}(\mathbf{w}_k|\mathbf{0}, \mathbf{Q}_k)$ and $p(\mathbf{v}_k) = \mathcal{N}(\mathbf{v}_k|\mathbf{0}, \mathbf{R}_k)$. The nonlinear mappings $\mathbf{f} : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_x}$ and $\mathbf{h} : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$ represent the process dynamics and the measurement model, respectively.

Define the measurement set $\mathbf{Y}_k := \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_k\}$. At time $k-1$, a statistical description of \mathbf{x}_k from \mathbf{Y}_{k-1} is given as $p(\mathbf{x}_k|\mathbf{Y}_{k-1})$. When the new measurement \mathbf{y}_k containing further information about \mathbf{x}_k arrives, $p(\mathbf{x}_k|\mathbf{Y}_{k-1})$ will be updated to $p(\mathbf{x}_k|\mathbf{Y}_k)$. Applying the Bayes' rule to this process, we have the following two-step Bayesian estimation paradigm that sequentially computes $p(\mathbf{x}_k|\mathbf{Y}_{k-1})$ and $p(\mathbf{x}_k|\mathbf{Y}_k)$:

- Prediction

$$p(\mathbf{x}_k|\mathbf{Y}_{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{Y}_{k-1})d\mathbf{x}_{k-1}, \quad (2)$$

- Update

$$p(\mathbf{x}_k|\mathbf{Y}_k) = \frac{p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{Y}_{k-1})}{p(\mathbf{y}_k|\mathbf{Y}_{k-1})}. \quad (3)$$

A key assumption to be made throughout the paper is that $p(\mathbf{x}_k|\mathbf{Y}_{k-1})$ and $p(\mathbf{y}_k|\mathbf{Y}_{k-1})$ are Gaussian. In this circumstance, $p(\mathbf{x}_k|\mathbf{Y}_k)$ is ensured to be Gaussian as well, and furthermore, the Bayesian filter in (2)-(3) propagates forward in a Gaussian manner. Accordingly, the Gaussian filtering equations can be obtained by determining the means and covariances of $p(\mathbf{x}_k|\mathbf{Y}_{k-1})$ and $p(\mathbf{x}_k|\mathbf{Y}_k)$. The prediction \mathbf{x}_k given \mathbf{Y}_{k-1} , denoted as $\hat{\mathbf{x}}_{k|k-1}$, is given by

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \int \mathbf{x}_k p(\mathbf{x}_k|\mathbf{Y}_{k-1})d\mathbf{x}_k \\ &= \int \left[\int \mathbf{x}_k p(\mathbf{x}_k|\mathbf{x}_{k-1})d\mathbf{x}_k \right] p(\mathbf{x}_{k-1}|\mathbf{y}_{k-1})d\mathbf{x}_{k-1} \\ &= \int \mathbf{f}(\mathbf{x}_{k-1}) \cdot \mathcal{N}(\mathbf{x}_{k-1}|\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}^x)d\mathbf{x}_{k-1}. \end{aligned} \quad (4)$$

The associated prediction error covariance is

$$\begin{aligned} \mathbf{P}_{k|k-1}^x &= \int (\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})^T p(\mathbf{x}_k|\mathbf{Y}_{k-1})d\mathbf{x}_k \\ &= \int \mathbf{x}_k \mathbf{x}_k^T p(\mathbf{x}_k|\mathbf{Y}_{k-1})d\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1} \hat{\mathbf{x}}_{k|k-1}^T \\ &= \int \mathbf{f}(\mathbf{x}_{k-1}) \mathbf{f}(\mathbf{x}_{k-1})^T \cdot \mathcal{N}(\mathbf{x}_{k-1}|\hat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1|k-1}^x)d\mathbf{x}_{k-1} \\ &\quad - \hat{\mathbf{x}}_{k|k-1} \hat{\mathbf{x}}_{k|k-1}^T + \mathbf{Q}_{k-1}. \end{aligned} \quad (5)$$

The derivation of (4)-(5) uses (A.1)-(A.2) in the Appendix.

When \mathbf{y}_k is available, let us consider the joint conditional pdf of \mathbf{x}_k and \mathbf{y}_k given \mathbf{Y}_{k-1} , which, according to the assumption, is Gaussian:

$$p(\mathbf{x}_k, \mathbf{y}_k|\mathbf{Y}_{k-1}) = \mathcal{N} \left(\begin{bmatrix} \mathbf{x}_k \\ \mathbf{y}_k \end{bmatrix} \middle| \begin{bmatrix} \hat{\mathbf{x}}_{k|k-1} \\ \hat{\mathbf{y}}_{k|k-1} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{k|k-1}^x & \mathbf{P}_{k|k-1}^{xy} \\ \left(\mathbf{P}_{k|k-1}^{xy} \right)^T & \mathbf{P}_{k|k-1}^y \end{bmatrix} \right). \quad (6)$$

Here, $\hat{\mathbf{y}}_{k|k-1}$ is the prediction of \mathbf{y}_k given \mathbf{Y}_{k-1} , given by

$$\hat{\mathbf{y}}_{k|k-1} = \int \mathbf{y}_k p(\mathbf{y}_k|\mathbf{Y}_{k-1})d\mathbf{y}_k. \quad (7)$$

It is noted that

$$\begin{aligned} p(\mathbf{y}_k|\mathbf{Y}_{k-1}) &= \int p(\mathbf{x}_k, \mathbf{y}_k|\mathbf{Y}_{k-1})d\mathbf{x}_k \\ &= \int p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{Y}_{k-1})d\mathbf{x}_k. \end{aligned}$$

Inserting the above equation into (7) yields

$$\begin{aligned} \hat{\mathbf{y}}_{k|k-1} &= \int \left[\int \mathbf{y}_k p(\mathbf{y}_k|\mathbf{x}_k)d\mathbf{y}_k \right] p(\mathbf{x}_k|\mathbf{Y}_{k-1})d\mathbf{x}_k \\ &= \int \mathbf{h}(\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{Y}_{k-1})d\mathbf{x}_k \\ &= \int \mathbf{h}(\mathbf{x}_k) \cdot \mathcal{N}(\mathbf{x}_k|\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}^x)d\mathbf{x}_k. \end{aligned} \quad (8)$$

The associated covariance is

$$\begin{aligned} \mathbf{P}_{k|k-1}^y &= \int (\mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1})(\mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1})^T p(\mathbf{y}_k|\mathbf{Y}_{k-1})d\mathbf{y}_k \\ &= \int \mathbf{h}(\mathbf{x}_k) \mathbf{h}^T(\mathbf{x}_k) \cdot \mathcal{N}(\mathbf{x}_k|\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}^x)d\mathbf{x}_k \\ &\quad - \hat{\mathbf{y}}_{k|k-1} \hat{\mathbf{y}}_{k|k-1}^T + \mathbf{R}_k, \end{aligned} \quad (9)$$

and the cross-covariance is

$$\begin{aligned} \mathbf{P}_{k|k-1}^{xy} &= \int \int (\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})(\mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1})^T \\ &\quad \cdot p(\mathbf{x}_k, \mathbf{y}_k|\mathbf{Y}_{k-1})d\mathbf{x}_k d\mathbf{y}_k \\ &= \int \mathbf{x}_k \mathbf{h}^T(\mathbf{x}_k) \cdot \mathcal{N}(\mathbf{x}_k|\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}^x)d\mathbf{x}_k \\ &\quad - \hat{\mathbf{x}}_{k|k-1} \hat{\mathbf{y}}_{k|k-1}^T. \end{aligned} \quad (10)$$

It follows from (6) and (A.3) that

$$p(\mathbf{x}_k|\mathbf{Y}_k) = \mathcal{N}(\mathbf{x}_k|\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}^x),$$

where

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{P}_{k|k-1}^{\mathbf{x}\mathbf{y}} \left(\mathbf{P}_{k|k-1}^{\mathbf{y}} \right)^{-1} (\mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1}), \quad (11)$$

$$\mathbf{P}_{k|k}^{\mathbf{x}} = \mathbf{P}_{k|k-1}^{\mathbf{x}} - \mathbf{P}_{k|k-1}^{\mathbf{x}\mathbf{y}} \left(\mathbf{P}_{k|k-1}^{\mathbf{y}} \right)^{-1} \left(\mathbf{P}_{k|k-1}^{\mathbf{x}\mathbf{y}} \right)^{\mathbf{T}}. \quad (12)$$

The Gaussian filter is summarized in Algorithm 1. It is noteworthy, however, that the Gaussian filter only delineates a conceptual framework for this type of filtering methods. To make it truly applicable in practice, it is needed to develop methods for evaluation of the integrals in (4)-(5) and (8)-(10).

Initialize: $k = 0$, $\hat{\mathbf{x}}_{0|0} = \mathbf{E}(\mathbf{x}_0)$, $\mathbf{P}_{0|0}^{\mathbf{x}} = p_0 \mathbf{I}$, where $p_0 > 0$
repeat
 $k \leftarrow k + 1$
 Prediction:
 State prediction via (4)
 Computation of prediction error covariance via (5)
 Update:
 Measurement prediction via (8) with the associated covariance via (9)
 Computation of the cross-covariance via (10)
 State estimation via (11)
 Computation of the estimation error covariance via (12)
until no more measurements arrive

Algorithm 1: The Gaussian filter for nonlinear state estimation.

III. RADIAL BASIS GAUSSIAN FILTERING

As is observed, the integrals in (4)-(5) and (8)-(10) take one of the following forms:

$$\Omega_1 = \int \mathbf{g}(\mathbf{x}) \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{x}, \quad (13)$$

$$\Omega_2 = \int \mathbf{x} \mathbf{g}^{\mathbf{T}}(\mathbf{x}) \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{x}, \quad (14)$$

$$\Omega_3 = \int \mathbf{g}(\mathbf{x}) \mathbf{g}^{\mathbf{T}}(\mathbf{x}) \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{x}. \quad (15)$$

where $\mathbf{x} \in \mathbb{R}^n$, and \mathbf{g} is assumed without loss of generality to be a mapping over $\mathbb{R}^n \rightarrow \mathbb{R}^m$, where n and m are an arbitrary positive integers. In this section, we introduce the notion and realization of the RBF approximation of $\mathbf{g}(\mathbf{x})$, and then continue to show how to construct the RB-GF, based on the proposed function approximation.

A. Filtering Integral Evaluation

For $g_i(\mathbf{x})$ that is the i -th element of $\mathbf{g}(\mathbf{x})$, we consider a nonlinearly parameterized approximator

$$\hat{g}_i(\mathbf{x}) = \sum_{j=1}^N w_{ij} s_j(\mathbf{x}), \quad (16)$$

where $s_j(\mathbf{x})$ for $j = 1, 2, \dots, N$ is a set of N RBFs and w_{ij} 's are the weighting factors. A wide variety of RBFs such as multi-quadratics, inverse multi-quadratics and Gaussian functions, have been studied in the literature. We propose to use the Gaussian RBFs (GRBFs), which will facilitate addressing the problem of Gaussian filtering. Then $s_j(\mathbf{x})$ is given by

$$\begin{aligned} s_j(\mathbf{x}) &= \exp \left[-\frac{(\mathbf{x} - \mathbf{c}_j)^{\mathbf{T}}(\mathbf{x} - \mathbf{c}_j)}{2\sigma_j^2} \right] \\ &= \alpha_j \cdot \mathcal{N}(\mathbf{x}|\mathbf{c}_j, \sigma_j^2 \mathbf{I}) \end{aligned}$$

where $\alpha_j = (2\pi\sigma_j^2)^{-\frac{n}{2}}$, \mathbf{c}_j and σ_j are the center and width of the RBF, respectively. For simplicity, we assume that \mathbf{c}_j 's and σ_j 's are fixed and known. The assumption does not limit the extension of the ensuing derivation to the case when both of them are unknown and need to be determined.

It follows from (A.4) that the multiplication of two Gaussian functions is another unnormalized Gaussian function. Hence, we consider

$$\begin{aligned} \beta_j \cdot \mathcal{N}(\mathbf{x}|\bar{\boldsymbol{\mu}}_j, \bar{\boldsymbol{\Sigma}}_j) &= \mathcal{N}(\mathbf{x}|\mathbf{c}_j, \sigma_j^2 \mathbf{I}) \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \\ \gamma_{jl} \cdot \mathcal{N}(\mathbf{x}|\underline{\boldsymbol{\mu}}_{jl}, \underline{\boldsymbol{\Sigma}}_{jl}) &= \mathcal{N}(\mathbf{x}|\mathbf{c}_j, \sigma_j^2 \mathbf{I}) \cdot \mathcal{N}(\mathbf{x}|\mathbf{c}_l, \sigma_l^2 \mathbf{I}) \\ &\quad \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \end{aligned}$$

where

$$\begin{aligned} \bar{\boldsymbol{\Sigma}}_j &= (\sigma_j^{-2} \mathbf{I} + \boldsymbol{\Sigma}^{-1})^{-1}, \\ \bar{\boldsymbol{\mu}}_j &= \bar{\boldsymbol{\Sigma}}_j (\sigma_j^{-2} \mathbf{c}_j + \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}), \\ \beta_j &= (2\pi)^{-\frac{n}{2}} \sigma_j^{-n} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} |\bar{\boldsymbol{\Sigma}}_j|^{\frac{1}{2}} \\ &\quad \cdot \exp \left[-\frac{1}{2} \left(\sigma_j^{-2} \mathbf{c}_j^{\mathbf{T}} \mathbf{c}_j + \boldsymbol{\mu}^{\mathbf{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} - \bar{\boldsymbol{\mu}}_j^{\mathbf{T}} \bar{\boldsymbol{\Sigma}}_j^{-1} \bar{\boldsymbol{\mu}}_j \right) \right], \\ \underline{\boldsymbol{\Sigma}}_{jl} &= (\sigma_l^{-2} \mathbf{I} + \bar{\boldsymbol{\Sigma}}_j^{-1})^{-1}, \\ \underline{\boldsymbol{\mu}}_{jl} &= \underline{\boldsymbol{\Sigma}}_{jl} (\sigma_l^{-2} \mathbf{c}_l + \bar{\boldsymbol{\Sigma}}_j^{-1} \bar{\boldsymbol{\mu}}_j), \\ \gamma_{jl} &= \beta_j (2\pi)^{-\frac{n}{2}} \sigma_l^{-n} |\bar{\boldsymbol{\Sigma}}_j|^{-\frac{1}{2}} |\underline{\boldsymbol{\Sigma}}_{jl}|^{\frac{1}{2}} \\ &\quad \cdot \exp \left[-\frac{1}{2} \left(\sigma_l^{-2} \mathbf{c}_l^{\mathbf{T}} \mathbf{c}_l + \bar{\boldsymbol{\mu}}_j^{\mathbf{T}} \bar{\boldsymbol{\Sigma}}_j^{-1} \bar{\boldsymbol{\mu}}_j - \underline{\boldsymbol{\mu}}_{jl}^{\mathbf{T}} \underline{\boldsymbol{\Sigma}}_{jl}^{-1} \underline{\boldsymbol{\mu}}_{jl} \right) \right]. \end{aligned}$$

Note that if $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are variables, $\bar{\boldsymbol{\Sigma}}_j$, $\bar{\boldsymbol{\mu}}_j$, β_j , $\underline{\boldsymbol{\Sigma}}_{jl}$, $\underline{\boldsymbol{\mu}}_{jl}$ and γ_{jl} are all functions of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. We have the following integration formulae before proceeding further:

$$\begin{aligned} \int s_j(\mathbf{x}) \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{x} &= \alpha_j \cdot \beta_j(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \\ \int \mathbf{x} s_j(\mathbf{x}) \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{x} &= \alpha_j \cdot \beta_j(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \cdot \bar{\boldsymbol{\mu}}_j(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \\ \int s_j(\mathbf{x}) s_l(\mathbf{x}) \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{x} &= \alpha_j \alpha_l \cdot \gamma_{jl}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \end{aligned}$$

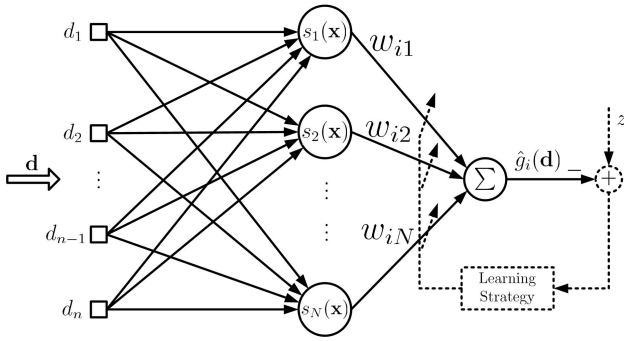


Fig. 1: The architecture of a RBFNN

Define the following matrices and vectors:

$$\mathbf{W} = \begin{bmatrix} \vdots & \vdots & \vdots \\ \cdots & w_{ij} & \cdots \\ \vdots & \vdots & \vdots \end{bmatrix},$$

$$\mathbf{s}(\mathbf{x}) = [\cdots \quad s_j(\mathbf{x}) \quad \cdots]^T,$$

$$\boldsymbol{\beta}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \begin{bmatrix} \vdots \\ \alpha_j \cdot \beta_j(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ \vdots \end{bmatrix},$$

$$\boldsymbol{\Psi}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = [\cdots \quad \alpha_j \cdot \beta_j(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \cdot \bar{\boldsymbol{\mu}}_j(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad \cdots],$$

$$\boldsymbol{\Gamma}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \begin{bmatrix} \vdots & \vdots & \vdots \\ \cdots & \alpha_j \alpha_l \cdot \gamma_{jl}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) & \cdots \\ \vdots & \vdots & \vdots \end{bmatrix}.$$

Here, $\mathbf{W} \in \mathbb{R}^{m \times N}$, $\mathbf{s} \in \mathbb{R}^N$, $\boldsymbol{\beta} \in \mathbb{R}^N$, $\boldsymbol{\Psi} \in \mathbb{R}^{N \times m}$ and $\boldsymbol{\Gamma} \in \mathbb{R}^{N \times N}$. We then have $\hat{\mathbf{g}}(\mathbf{x}) = \mathbf{W} \cdot \mathbf{s}(\mathbf{x})$ and

$$\begin{aligned} \Omega_1 &\approx \int \hat{\mathbf{g}}(\mathbf{x}) \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{x} \\ &= \mathbf{W} \cdot \boldsymbol{\beta}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \end{aligned} \quad (17)$$

$$\begin{aligned} \Omega_2 &\approx \int \mathbf{x} \hat{\mathbf{g}}^T(\mathbf{x}) \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{x} \\ &= \boldsymbol{\Psi}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \cdot \mathbf{W}^T, \end{aligned} \quad (18)$$

$$\begin{aligned} \Omega_3 &\approx \int \hat{\mathbf{g}}(\mathbf{x}) \hat{\mathbf{g}}^T(\mathbf{x}) \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\mathbf{x} \\ &= \mathbf{W} \cdot \boldsymbol{\Gamma}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \cdot \mathbf{W}^T. \end{aligned} \quad (19)$$

We see that (17)-(19) construct a computational foundation, on which the Gaussian filtering integrals in (4)-(5) and (8)-(10) can be evaluated easily. The foundation is based on RBF approximation. If the approximation is accurate, (17)-(19) will be a closed-form solution to Gaussian filtering.

B. RBF Approximation via Neural Network Learning

Prior to integral evaluation in (17)-(19), the weight matrix \mathbf{W} must be determined optimally in the sense that approximation error between $\mathbf{g}(\mathbf{x})$ and $\hat{\mathbf{g}}(\mathbf{x})$ is minimized. A formulation of this problem is: Given a data set containing M elements, $\mathcal{D} = \{(\mathbf{d}_j, \mathbf{z}_j) | \mathbf{z}_j = \mathbf{g}(\mathbf{d}_j), \mathbf{d}_j \in \mathbb{R}^n, \mathbf{z}_j \in$

$\mathbb{R}^m, j = 1, 2, \dots, M\}$, find the optimal \mathbf{W} to minimize the convex cost function $J(\mathbf{W})$ defined as

$$\begin{aligned} J(\mathbf{W}) &= \frac{1}{2} \sum_{j=1}^M \|\mathbf{z}_j - \hat{\mathbf{g}}(\mathbf{d}_j)\|^2 \\ &= \frac{1}{2} \sum_{j=1}^M \|\mathbf{z}_j - \mathbf{W} \cdot \mathbf{s}(\mathbf{d}_j)\|^2 \\ &= \frac{1}{2} \sum_{j=1}^M \sum_{i=1}^m \|z_{ji} - \mathbf{w}_i \cdot \mathbf{s}(\mathbf{d}_j)\|^2, \end{aligned} \quad (20)$$

where z_{ji} is the i -th element of \mathbf{z}_j and \mathbf{w}_i is the i -th row vector of \mathbf{W} . It is noteworthy that each \mathbf{w}_i can be determined separately by minimizing

$$J(\mathbf{w}_i) = \sum_{j=1}^M \|z_{ji} - \mathbf{w}_i \cdot \mathbf{s}(\mathbf{d}_j)\|^2.$$

The above weight determination problem is equivalent to training a RBF Neural Network (RBFNN). A RBFNN usually performs curve fitting in a high dimensional space, or more specifically, to find a hypersurface that provides a best fit for the high-dimensional training data [10].

For $g_i(\mathbf{x})$, the schematic diagram of a RBFNN is shown in Fig. 1. It has three layers. The first one is the input layer, which has n nodes corresponding to each element of the input vector \mathbf{d} . The second layer is a hidden layer with N units, to each of which all nodes in the first layer are connected. The activation functions of the j -th unit is the GRBF $s_j(\mathbf{x})$, indicating that this is indeed a GRBFNN. Each $s_j(\mathbf{x})$ in the hidden layer is connected through the weight w_{ij} to the output layer that has only a single unit. This unit computes a weighted sum of the outputs of the hidden units as the output of the network.

It is noted that the RBFNN translates the function approximation under consideration into neural network learning, which applies learning strategies to the training data set \mathcal{D} to determine the weights of the output layer. A few different types of learning strategies have been proposed in the literature. A straightforward approach is to use the pseudoinverse method to derive the least squares solution to (20). However, it is computationally inefficient, especially whenever new data become available, and also poorly scalable to large data sets. To remedy this situation, most other approaches for RBFNN learning carry out recursive updating. Among them, we highlight the one based on gradient descent [10]. Consider \hat{g}_i in (16), which can be rewritten as $\hat{g}_i(\mathbf{x}) = \mathbf{w}_i \cdot \mathbf{s}(\mathbf{x})$. The recursive learning procedure of \mathbf{w}_i is expressed as

$$\phi(\ell) = \sum_{j=1}^M \mathbf{s}^T(\mathbf{d}_j) (z_{ji} - \mathbf{w}_i(\ell) \cdot \mathbf{s}(\mathbf{d}_j)), \quad (21)$$

$$\mathbf{w}_i(\ell+1) = \mathbf{w}_i(\ell) - \eta \cdot \phi(\ell), \quad (22)$$

where $\phi = \nabla_{\mathbf{w}_i} J(\mathbf{w}_i)$ is the gradient, η is the learning coefficient and ℓ denotes the recursion step.

Approximation properties of the RBFNN is of much significance in practical implementation. The *Universal Approximation Theorem* states that, if $\mathbf{g}(\mathbf{x})$ is continuous, then there is a RBFNN such that the function $\hat{\mathbf{g}}(\mathbf{x})$ realized by the RBFNN is close to $\mathbf{g}(\mathbf{x})$ in the L_p norm for $p \in [1, \infty]$ [10]. Furthermore, it is pointed out in [12] that $\hat{\mathbf{g}}(\mathbf{x})$ can approximate the continuous $\mathbf{g}(\mathbf{x})$ to an arbitrary accuracy over a compact set. Thus for the considered Gaussian filtering problem, if the function approximators are well designed via RBFNNs, high-accuracy approximation can be achieved, thus ensuring the filtering performance.

C. The RB-GF

Putting together the formulae in the Gaussian filter, function approximation and integral evaluation yields the RB-GF, as described in Algorithm 2.

Initialize: $k = 0$, $\hat{\mathbf{x}}_{0|0} = \mathbf{E}(\mathbf{x}_0)$, $\mathbf{P}_{0|0}^{\mathbf{x}} = p_0 \mathbf{I}$, where $p_0 > 0$, typically a large positive value

Function approximation
Construction of the RBF based approximators for \mathbf{f} and \mathbf{h} using RBFNN via (21)-(22)

repeat
 $k \leftarrow k + 1$
Prediction:
 State prediction via (4) and (17)
 Computation of prediction error covariance via (5) and (19)
Update:
 Measurement prediction via (8) and (17) with the associated covariance via (9) and (19)
 Computation of the cross-covariance via (10) and (18)
 State estimation via (11)
 Computation of the estimation error covariance via (12)
until no more measurements arrive

Algorithm 2: The Radial Basis Gaussian Filter (RB-GF).

IV. SIMULATION EXAMPLE

In this section, we present a numerical example to evaluate the performance of the proposed RB-GF. Consider the one-dimensional nonlinear system

$$\begin{cases} x_{k+1} = f(x_k) + w_k, \\ y_k = h(x_k) + v_k, \end{cases} \quad (23)$$

where

$$\begin{aligned} f(x_k) &= \sin(x_k) + 0.05x_k(1 - x_k^2), \\ h(x_k) &= 0.01(x_k - 0.05). \end{aligned}$$

The noises w_k and v_k are zero-mean white Gaussian with $Q = 0.001$ and $R = 0.001$, respectively. We then apply the RB-GF algorithm to data generated from the above system. Another three types of Gaussian filters, UKF, GHF, CKF, are also implemented for an overall comparison. In simulation,

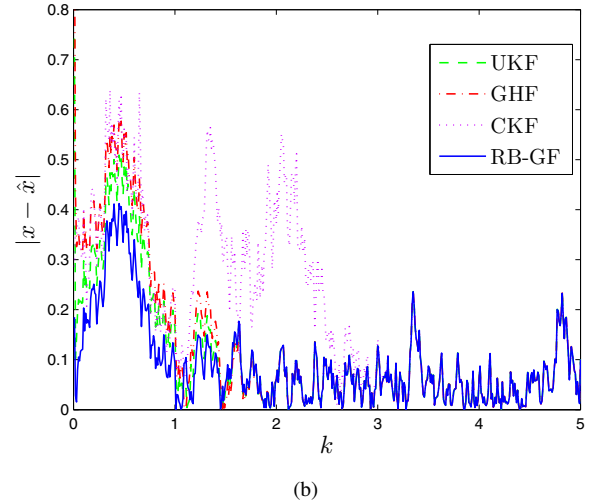
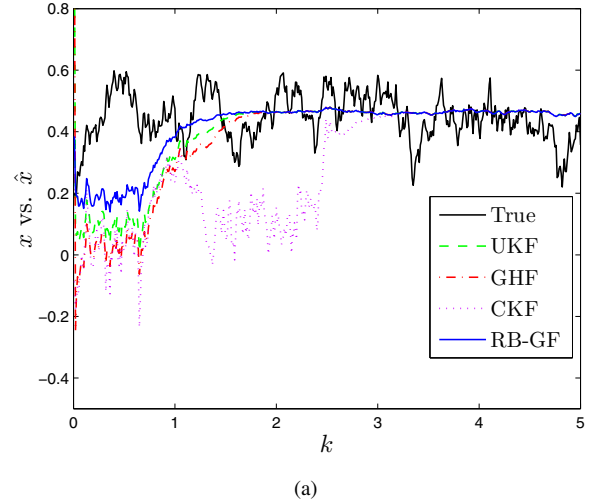


Fig. 2: (a) True states and estimated ones; (b) estimation errors.

the initial condition $x_0 = 0.2$, $\hat{x}_{0|0} = 0.6$ and $P_{0|0} = 3$. The functions $f(x)$ and $h(x)$ are approximated over $[-2, 2]$ by 10 GRBFs with the centers evenly distributed and widths equal to 1. The number of Sigma points in the UKF is 11 and the number of quadrature points in the GHF is 10.

Fig. 2 shows the state estimation results yielded by the four filters. From Fig. 2 and numerous simulation runs, we consistently observe comparable performance between the UKF, GHF and the proposed RB-GF, with the RB-GF usually performing slightly better. Although the CKF can achieve the same-level accuracy only when a sufficient number of measurements arrive, it is the most computationally efficient.

From the simulation, we gain an insight that the RB-GF can be enhanced in two ways. First, instead of designing the approximators by RBFNN learning ‘once and for all’, dynamically approximating the functions over the neighborhood of the current state estimate will lead to better approximation performance. Second, the notion developed

in this paper can be extended to mixed Gaussian filter, which employs a linear combination of Gaussian densities to represent any type of probability densities. Despite additional computational complexity, both methods are foreseeable to boost the estimation performance significantly.

V. CONCLUSIONS

We have investigated the nonlinear state estimation problem and proposed a new Gaussian filter based on RBF approximation. A distinct advantage of Gaussian filters is the closed-form description. However, their practical implementation requires evaluation of certain forms of integrals. To deal with this challenge, we have proposed to build approximators composed of a weighed sum of RBFs for nonlinear system functions. Determination of the approximators, i.e., optimal selection of the weights, can be addressed by RBFNN learning. It has been shown that, with the RBF based function approximators, the integrals in Gaussian filtering can be delicately evaluated, giving rise to the RB-GF algorithm. We have demonstrated the effectiveness of the RB-GF through a comparison with other filters in numerical simulation. Future work will be devoted to filtering performance enhancement by developing dynamic function approximation and the mixed Gaussian filter with RBF based function approximators.

APPENDIX

For the reader's convenience, we provide some Gaussian identities, the proofs of which are widely available in textbooks and thus omitted. All the vectors and matrices involved below are assumed to have compatible dimensions.

- If $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A})$, then

$$\int (\mathbf{M}\mathbf{x} + \mathbf{m})p(\mathbf{x})d\mathbf{x} = \mathbf{M}\mathbf{a} + \mathbf{m}, \quad (\text{A.1})$$

$$\int (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T p(\mathbf{x})d\mathbf{x} = (\mathbf{a} - \mathbf{m})(\mathbf{a} - \mathbf{m})^T + \mathbf{A}. \quad (\text{A.2})$$

- If

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \middle| \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{B} \end{bmatrix}\right),$$

then the marginal distributions are $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A})$ and $p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{b}, \mathbf{B})$, and the conditional distributions is

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\mathbf{a} + \mathbf{C}\mathbf{B}^{-1}(\mathbf{y} - \mathbf{b}), \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^T). \quad (\text{A.3})$$

- Given two gaussian functions, their multiplication leads to another gaussian function, i.e.,

$$\mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A}) \cdot \mathcal{N}(\mathbf{x}|\mathbf{b}, \mathbf{B}) = \lambda \cdot \mathcal{N}(\mathbf{x}|\mathbf{c}, \mathbf{C}), \quad (\text{A.4})$$

where

$$\mathbf{C} = (\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1},$$

$$\mathbf{c} = \mathbf{C}(\mathbf{A}^{-1}\mathbf{a} + \mathbf{B}^{-1}\mathbf{b}),$$

$$\lambda = (2\pi)^{-\frac{n}{2}} |\mathbf{A}|^{-\frac{1}{2}} |\mathbf{B}|^{-\frac{1}{2}} |\mathbf{C}|^{\frac{1}{2}}$$

$$\cdot \exp\left[-\frac{1}{2}(\mathbf{a}^T \mathbf{A}^{-1} \mathbf{a} + \mathbf{b}^T \mathbf{B}^{-1} \mathbf{b} - \mathbf{c}^T \mathbf{C}^{-1} \mathbf{c})\right].$$

REFERENCES

- [1] B. D. O. Anderson and J. B. Moore, *Optimal filtering*. Prentice-Hall, Englewood Cliffs, 1979.
- [2] D. Simon, *Optimal State Estimation: Kalman, H_∞ , and Nonlinear Approaches*. Wiley-Interscience, 2006.
- [3] J. V. Candy, *Bayesian Signal Processing: Classical, Modern and Particle Filtering Methods*. New York, NY, USA: Wiley-Interscience, 2009.
- [4] P. S. Maybeck, *Stochastic models, estimation and control, Volume II*, A. Press, Ed., 1979.
- [5] G. Evensen, "Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics," *Journal of Geophysical Research*, vol. 99, pp. 10 143–10 162, 1994.
- [6] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.
- [7] H. Kushner and A. Budhiraja, "A nonlinear filtering algorithm based on an approximation of the conditional distribution," *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 580–585, 2000.
- [8] K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *IEEE Transactions on Automatic Control*, vol. 45, no. 5, pp. 910–927, 2000.
- [9] I. Arasaratnam and S. Haykin, "Cubature Kalman filters," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, 2009.
- [10] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.
- [11] S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive Neural Network Control for Robotic Manipulators*. London, UK: World Scientific, 1998.
- [12] R. Sanner and J.-J. Slotine, "Gaussian networks for direct adaptive control," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 837–863, 1992.
- [13] B. Ren, S. Ge, T. H. Lee, and C.-Y. Su, "Adaptive neural control for a class of nonlinear systems with uncertain hysteresis inputs and time-varying state delays," *IEEE Transactions on Neural Networks*, vol. 20, no. 7, pp. 1148–1164, 2009.
- [14] B. Ren, S. S. Ge, K. P. Tee, and T. H. Lee, "Adaptive neural control for output feedback nonlinear systems using a barrier Lyapunov function," *IEEE Transactions on Neural Networks*, vol. 21, no. 8, pp. 1339–1345, 2010.
- [15] S. Elanayar V.T. and Y. Shin, "Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems," *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 594–603, 1994.