KNN-Based Kalman Filter: An Efficient and Non-stationary Method for Gaussian Process Regression

Yali Wang

Shenzhen Institutes of Advanced Technology,
Chinese Academy of Sciences, China
Email: yl.wang@siat.ac.cn

Brahim Chaib-draa

Department of Computer Science and Software Engineering,
Laval University, Canada
Email: chaib@ift.ulaval.ca

Abstract

The traditional Gaussian process (GP) regression is often deteriorated when the data set is large-scale and/or non-stationary. To address these challenging data properties, we propose a K-Nearest-Neighbor-based Kalman filter for Gaussian process regression (KNN-KFGP). Firstly, we design a test-input-driven KNN mechanism to group the training set into a number of small collections. Secondly, we use the latent function values of these collections as the unknown states and then construct a novel state space model with GP prior. Thirdly, we explore Kalman filter on this state space model to efficiently filter out the latent function values for prediction. As a result, our KNN-KFGP framework can effectively alleviate the heavy computation load of GP with recursive Bayesian inference, especially when the data set is large-scale. Moreover, our KNN mechanism helps each test point to find its strongly-correlated local training subset, and thus our KNN-KFGP can model non-stationarity in a flexible manner. Finally, we compare our KNN-KFGP to several related works and show its superior performance on a number of synthetic and real-world data sets.

Keywords: Gaussian Process Regression; K-Nearest-Neighbor (KNN); Kalman Filter; Online Learning; Non-stationarity.

---

1. Introduction

Over the past years, Bayesian parametric models have been well-developed for regression [1, 2]. However, these models are often limited to capture data flexibility in all realistic details. Hence, Bayesian nonparametric methods have become popular. In particular, Gaussian Process (GP) [3], a powerful nonparametric model with an elegant inference framework, has been successfully applied to a number of real-world regression tasks in robotics [4, 5, 6, 7], geophysics [8, 9], econometrics [10, 11] and so on. However, the performance of GP is often deteriorated, when the data set is large-scale [12, 13, 14] and/or exhibits non-stationarity [15, 16, 17].

1.1. GP Variants for Large-scale Data

For large-scale data sets, the computation of GP is expensive $O(N^3)$, where N is the size of training set. To reduce computation, several GP variants have been proposed and a detailed review can be found in [12, 13, 18]. In this paper, we mainly discuss several related approaches, according to different mechanisms for computation reduction.

One well-known mechanism is sparse approximation such as sparse pseudo-input Gaussian process (SPGP) [19] and sparse spectrum Gaussian process (SSGP) [20]. In SPGP, sparsification is achieved by using a small number of learned pseudo-inputs to parameterize the covariance matrix of training set. However, its accuracy is often limited, especially when the size of the pseudo set is getting larger and/or the dimension of the input space is getting higher [19]. In SSGP, sparsification is achieved by learning a stationary trigonometric Bayesian model. But similar to SPGP, SSGP may trap into overfitting as optimization is implemented over a large number of model parameters.

Another important mechanism is online-learning GP variants [21, 22, 23] which process the large-scale training set in a sequential manner to maintain computation efficiency. A well-known method is the recent Kalman filter Gaussian process (KFGP) [23]. By taking advantage of the novel connection between GP and Kalman filter, KFGP reduces computation by correlating GP of different training subsets in a recursive Bayesian filtering framework. However, the training subset at each iteration of KFGP is randomly selected, hence its accuracy is still limited, especially when the data sets exhibit non-stationarity.

1.2. GP Variants for Non-stationary Data

Many real-world applications in geophysics, biology and robotics often reflect strong non-stationarity [8, 24, 25, 26, 27]. It reflects an input-dependent smoothness phenomenon, where the correlation between the latent function values $f(\mathbf{x})$ and $f(\mathbf{x}')$ does not only depend on $\mathbf{x} - \mathbf{x}'$, but it is also related to the locations of inputs $\mathbf{x}$ and $\mathbf{x}'$ [3, 16, 25].

The traditional GP often mistakenly interpret it with the stationary assumption. A straightforward method for non-stationarity is to pre-define a non-stationary covariance function for GP [3, 8], or to warp GP with different nonlinear functions [28, 25, 29]. However, the performance of this method is restricted, due to the fact that the pre-defined functions require either prior knowledge on the specific non-stationarity [3, 25] or computationally-expensive inference mechanisms for model parameter learning [8, 29].

One popular mechanism for non-stationarity is the local-expert GP approach [24, 30, 31]. A well-known method is called the infinite mixture of Gaussian process experts (iMGPE) [24], where an input-dependent Dirichlet process is designed as a gating network for GP expert selection. However, the hierarchical model structure is difficult to implement in practice [8]. Several practical variants were proposed by constructing a local mixture of GP experts for large-scale computer vision and robotics applications [30, 31, 32], but a tradeoff between accuracy and efficiency has to be considered.

1.3. Our Contributions

Inspired by these variants of GP, we propose a novel K-Nearest-Neighbor-based Kalman filter for Gaussian process regression (KNN-KFGP) in this paper, to address both large-scale and non-stationary data properties within a unified, efficient and accurate Bayesian filtering framework.

Firstly, we design a test-input-driven KNN search mechanism to construct a small training collection for each test. *Then*, we take advantage of GP to correlate different KNN-based training collections in a novel state space model, where the states are the latent function values of these data collections. *Next*, we perform Kalman filter to infer the latent function values in an efficient predict-update manner.

Compared to KFGP [23], the non-trivial KNN mechanism in our approach can successfully discover a strongly-correlated training subset for each test. Hence, our KNN-KFGP is flexible to capture non-stationarity. Compared to the local-expert approach [24, 30, 31], our KNN-KFGP is based on an efficient Bayesian filtering framework to maintain computation load while preserving accuracy.

The rest of this paper is organized as follows. In Section 2, we briefly review GP and how to use it for regression. In Section 3, we explain how to design our KNN-KFGP framework in detail. In Section 4, we evaluate our proposed KNN-KFGP on several data sets and show its validity. Finally, we conclude the paper in Section 5.

2. Review of Gaussian Process

In this section, we first introduce the basics of Gaussian process (GP) and then explain how to use GP to address the regression task.

2.1. Gaussian Process

A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution [3]. It has been widely used as a prior over the latent function, where any finite number of the function values in this scenario are Gaussian-distributed random variables [1, 2, 3, 33].

Specifically, a GP prior over the latent function $f(\mathbf{x}) \in R$ is defined as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{1}$$

where $\mathbf{x}$, $\mathbf{x}' \in R^{D_x}$ are any two D_x-dimension input vectors, $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ are respectively the mean function and the covariance function [3],

$$\begin{aligned} m(\mathbf{x}) &= E[f(\mathbf{x})], & (2) \\ k(\mathbf{x}, \mathbf{x}') &= E[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. & (3) \end{aligned}$$

According to this definition, one can obtain any n latent function values $f(X) = \{f(\mathbf{x}_1), \cdots, f(\mathbf{x}_n)\}$ at the input vectors $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ are Gaussian-distributed random variables [3],

$$f(X) \sim \mathcal{N}(\mathbf{m}(X), K(X, X)), \tag{4}$$

where the mean vector $\mathbf{m}(X)$ is

$$\mathbf{m}(X) = \begin{bmatrix} m(\mathbf{x}_1) \\ \cdots \\ m(\mathbf{x}_n) \end{bmatrix}, \tag{5}$$

and the covariance matrix $K(X,X)$ is

$$K(X,X) = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & \cdots & k(\mathbf{x}_1,\mathbf{x}_n) \\ \cdots & \cdots & \cdots \\ k(\mathbf{x}_n,\mathbf{x}_1) & \cdots & k(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix}. \tag{6}$$

2.2. Gaussian Process Regression

After we introduce the basics of GP, we explain how to use the properties of GP to solve the standard regression task from a Bayesian perspective. Suppose that we are given a training set with N input ($\mathbf{x} \in R^{D_x}$) - output ($y \in R$) pairs, i.e., $(X,\mathbf{y}) = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$. Each output is assumed to be generated from

$$y = f(\mathbf{x}) + \epsilon, \tag{7}$$

where the noise ϵ is Gaussian $\epsilon \sim \mathcal{N}(0,\sigma^2)$ with variance σ^2. The goal of regression is to learn the latent function $f(\mathbf{x})$ with the training set $(X,\mathbf{y})$ and make prediction for new test inputs $X_\star = \{\mathbf{x}_\star^i\}_{i=1}^M$.

Since parametric approaches restrict the richness of the assumed function family when solving the regression problem, Bayesian nonparametric methods have recently become more attractive by giving a prior probability to every possible function [1]. Following this direction, we choose a widely-used GP prior [3] in this work, $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x},\mathbf{x}'))$, where the mean function $m(\mathbf{x})$ in Eq. (1) is zero [1], the covariance function $k(\mathbf{x},\mathbf{x}')$ is the squared exponential covariance function

$$k(\mathbf{x},\mathbf{x}') = \sigma_f^2 e^{-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^T L^{-2}(\mathbf{x}-\mathbf{x}')} \tag{8}$$

where $L = diag([l_1 \cdots l_d])$ is the length-scale matrix and σ_f^2 is the amplitude parameters. For convenience, the noise variance and all the GP hyperparameters are collected into a vector $\theta = [\sigma^2, \sigma_f^2, l_1, \cdots, l_d]^T$.

Given the training set $(X,\mathbf{y})$ and test inputs $X_\star$, one can take advantage of GP to address regression in a Bayesian manner. Specifically, the regression task can be interpreted to learn the predictive distribution over the test output vector $\mathbf{y}_\star$ and the parameter vector θ,

$$p(\mathbf{y}_\star, \theta | X_\star, X, \mathbf{y}) = p(\mathbf{y}_\star | X_\star, X, \mathbf{y}, \theta) p(\theta | X, \mathbf{y}). \tag{9}$$

[1]Note that it is straightforward to choose other mean functions to do mathematical derivations without difficulties.

Due to the fact that $p(\theta|X, \mathbf{y}) \propto p(\mathbf{y}|X, \theta)$, a popular approach to infer θ is to minimizing the negative log likelihood with gradient optimization [3]

$$\begin{aligned} & -\log p(\mathbf{y}|X, \theta) \\ =& \frac{1}{2}\mathbf{y}^T(K(X,X) + \sigma^2 I)^{-1}\mathbf{y} + \frac{1}{2}\log|K(X,X) + \sigma^2 I| + \frac{n}{2}\log 2\pi. \end{aligned} \tag{10}$$

In practice, this approach works well [3] and thus we apply it in this paper.

After θ is learned, we make the prediction for the test input set $X_\star$. Specifically, due to the fact that $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$ and the noise in Eq. (7) is Gaussian, $p(\mathbf{y}, f(X_\star)|X, X_\star, \theta)$ is jointly Gaussian,

$$p(\mathbf{y}, f(X_\star)|X, X_\star, \theta) = \mathcal{N}(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma^2 I & K(X, X_\star) \\ K(X_\star, X) & K(X_\star, X_\star) \end{bmatrix}), \tag{11}$$

where $f(X_\star)$ is the vector of latent function values at the test inputs $X_\star$. Then, based on the conditional property of Gaussian distribution, one can obtain that $p(f(X_\star)|X, \mathbf{y}, X_\star, \theta)$ is also Gaussian with the mean vector $\mu_\star$ and the covariance matrix $\Sigma_\star$ [3]

$$\mu_\star = K(X_\star, X)[K(X,X) + \sigma^2 I]^{-1}\mathbf{y}, \tag{12}$$

$$\Sigma_\star = K(X_\star, X_\star) - K(X_\star, X)[K(X,X) + \sigma^2 I]^{-1}K(X_\star, X)^T, \tag{13}$$

where $K(X,X)$ is constructed by Eq. (6), $K(X_\star, X)$ and $K(X_\star, X_\star)$ are constructed in the same way. Finally, the predictive distribution of the test output is also Gaussian, i.e., $p(\mathbf{y}_\star|X_\star, X, \mathbf{y}, \theta) = \mathcal{N}(\mu_\star, \Sigma_\star + \sigma^2 I)$, due to the Gaussian noise in Eq. (7).

The computation complexity of GP is mainly governed by the inversion of covariance matrix of N training pairs ($O(N^3)$) [3]. As a result, GP suffers from an unsatisfactory computation burden in practice, when N is beyond a few thousand. Furthermore, the performance of the standard GP regression is often deteriorated, when the data set exhibits non-stationarity [3, 8, 31]. To address these difficulties, we propose a novel KNN-Based Kalman Filter for GP regression in the following.

3. KNN-Based Kalman Filter for Gaussian Process Regression

In this section, we propose a novel KNN-based Kalman filter for Gaussian process regression (KNN-KFGP) in order to deal with the large-scale and/or non-stationary data sets in the real world. We firstly apply a test-input-driven KNN search to construct a small training subset for each test, and then establish a state space model by correlating different training subsets using the GP prior. Finally, Kalman filter is performed on our state space model to infer the latent function values in a recursive Bayesian manner.

3.1. State Space Model: Using GP to Correlate KNN-Based Data Collections

To capture non-stationarity when making prediction at the test inputs $\{\mathbf{x}_\star^i\}_{i=1}^M$, we propose a test-input-driven KNN search mechanism to explore the strongly-correlated local structure for each test. Concretely, for the test input $\mathbf{x}_\star^t$, we find its K_t nearest training inputs X_t according to the Euclidian distance. Then, we use X_t and the corresponding outputs $\mathbf{y}_t$ as a small training collection $(X_t, \mathbf{y}_t) = \{(\mathbf{x}_i, y_i)\}_{i=1}^{K_t}$ for $\mathbf{x}_\star^t$.

Next, we take advantage of GP to construct a novel state space model containing the *state* and *observation* equation, to process the large-scale data set sequentially and thus reduce the computation burden of GP regression. For clarity, we collect the training and test inputs of the t_{th} data collections into an input set $X_t^c = \{X_t, \mathbf{x}_\star^t\}$, and denote the latent function values of X_t^c as $f_t^c = f(X_t^c) = \{f(X_t), f(\mathbf{x}_\star^t)\}$. Additionally, we learn the model parameter vector θ in advance, by minimizing the negative log likelihood (Eq. (10)) with gradient optimization in Subsection 2.2. In this case, we omit θ in the following derivations without loss of generality.

Since our main task is the regression problem, the states in our state space model are the latent function values of the KNN-based data collections. Consequently, the *state* equation describes a Markovian transition of latent function values, where we use GP to explore the correlations between the $(t-1)_{th}$ and t_{th} data collections, i.e., f_{t-1}^c and f_t^c. By assuming $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$, we can obtain that $p(f_t^c, f_{t-1}^c | X_{t-1}^c, X_t^c)$ is Gaussian according to the definition of GP in Subsection 2.1,

$$p(f_t^c, f_{t-1}^c | X_{t-1}^c, X_t^c) = \mathcal{N}(\mathbf{0}, \begin{bmatrix} K(X_t^c, X_t^c) & K(X_t^c, X_{t-1}^c) \\ K(X_t^c, X_{t-1}^c)^T & K(X_{t-1}^c, X_{t-1}^c) \end{bmatrix}). \tag{14}$$

Then, due to the fact that the joint distribution $p(f_t^c, f_{t-1}^c | X_{t-1}^c, X_t^c)$ is Gaussian with the form in Eq. (14), we can obtain the following conditional distribution which is also Gaussian [3],

$$p(f_t^c | f_{t-1}^c, X_{t-1}^c, X_t^c) = \mathcal{N}(G_t f_{t-1}^c, Q_t), \tag{15}$$

with the transition matrix G_t and covariance matrix Q_t

$$G_t = K(X_t^c, X_{t-1}^c)K^{-1}(X_{t-1}^c, X_{t-1}^c), \tag{16}$$
$$Q_t = K(X_t^c, X_t^c) - K(X_t^c, X_{t-1}^c)K^{-1}(X_{t-1}^c, X_{t-1}^c)K(X_t^c, X_{t-1}^c)^T. \tag{17}$$

Next, based on Bayesian filtering theory [34], the conditional distribution in Eq. (15) is equivalent to the following *state* equation which is a linear transition between f_t^c and f_{t-1}^c with an additive Gaussian noise v_t^f,

$$f_t^c = G_t f_{t-1}^c + v_t^f, \quad v_t^f \sim \mathcal{N}(\mathbf{0}, Q_t). \tag{18}$$

The *observation* equation describes the relation between the hidden function values and their noisy outputs. It can be directly obtained from Eq. (7),

$$\mathbf{y}_t = H_t f_t^c + v_t^y, \quad v_t^y \sim \mathcal{N}(\mathbf{0}, R_t), \tag{19}$$

where $H_t = [I_{K_t} \;\; \mathbf{0}]$ is an index matrix to make sure $H_t f_t^c = f(X_t)$. It is due to the fact that only the training inputs X_t have the corresponding outputs $\mathbf{y}_t$. Furthermore, the noise v_t^y is from ϵ in (7) with $R_t = \sigma^2 I$.

To sum up, our state space model is fully specified by Eq. (18)-(19). The whole construction is shown in Fig. 1. Next, we take advantage of our state space model to transform regression into a state estimation problem, where we perform Kalman Filter to estimate $f_t^c = f(X_t^c) = \{f(X_t), f(\mathbf{x}_\star^t)\}$ in a recursive fashion, by using the training outputs $\mathbf{y}_t$ of the t-th training collection $(X_t, \mathbf{y}_t)$.

3.2. Bayesian Inference: Kalman Filter

According to Eq. (18)-(19), we can see that our state space model is linear with additive Gaussian noises. In this case, we propose to perform the well-known Kalman Filter (KF), which is an optimal solution by maximum-a-posteriori (MAP) [35], to infer the latent function values. The Bayesian inference procedure of KF is based on a *predict-update* fashion. Due to linear and Gaussian properties in our state space model, all the distributions involved in KF are Gaussian [35]. Suppose that the $(t-1)_{th}$ posterior distribution is

$$p(f_{t-1}^c | \mathbf{y}_{1:t-1}, X_{1:t-1}^c) = \mathcal{N}(\mu_{t-1|t-1}^c, \Sigma_{t-1|t-1}^c), \tag{20}$$

with the mean vector $\mu_{t-1|t-1}^c$ and covariance matrix $\Sigma_{t-1|t-1}^c$.

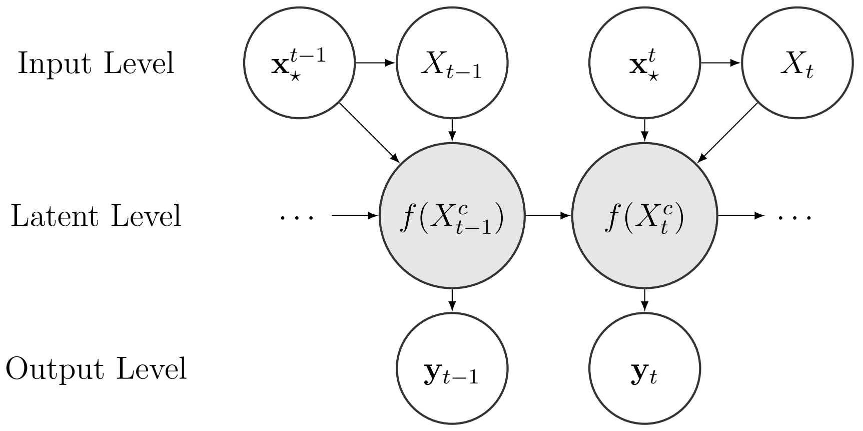


Figure 1: State space model construction by using GP to correlate the KNN-based data collections. The arrows between $\mathbf{x}_\star^{t-1}$ and X_{t-1}, $\mathbf{x}_\star^{t}$ and X_t represent the test-input-driven KNN mechanism. $f(X_t^c) = \{f(X_t), f(\mathbf{x}_\star^t)\}$ are the latent function values of $X_t^c = \{X_t, \mathbf{x}_\star^t\}$.

The *predict* step is to incorporate the state transition $p(f_t^c|f_{t-1}^c, X_{t-1}^c, X_t^c)$ (directly from Eq. (15)) to predict the t_{th} hidden function values,

$$\begin{aligned}
&p(f_t^c|\mathbf{y}_{1:t-1}, X_{1:t}^c)\\
=&\int p(f_t^c|f_{t-1}^c, X_{t-1}^c, X_t^c)p(f_{t-1}^c|\mathbf{y}_{1:t-1}, X_{1:t-1}^c)df_{t-1}^c\\
=&\int \mathcal{N}(G_t f_{t-1}^c, Q_t)\mathcal{N}(\mu_{t-1|t-1}^c, \Sigma_{t-1|t-1}^c)df_{t-1}^c\\
=&\mathcal{N}(G_t\mu_{t-1|t-1}^c, \quad G_t\Sigma_{t-1|t-1}^c G_t^T + Q_t).
\end{aligned} \tag{21}$$

We denote $p(f_t^c|\mathbf{y}_{1:t-1}, X_{1:t}^c)$ as $\mathcal{N}(\mu_{t|t-1}^c, \Sigma_{t|t-1}^c)$, then the *predict* step is fully expressed by

$$\mu_{t|t-1}^c = G_t\mu_{t-1|t-1}^c, \tag{22}$$

$$\Sigma_{t|t-1}^c = G_t\Sigma_{t-1|t-1}^c G_t^T + Q_t. \tag{23}$$

The *update* step is to take advantage of Bayes rule to update the prediction with the t_{th} observed output $\mathbf{y}_t$,

$$p(f_t^c|\mathbf{y}_{1:t}, X_{1:t}^c) = \frac{p(\mathbf{y}_t|f_t^c, X_t^c)p(f_t^c|\mathbf{y}_{1:t-1}, X_{1:t}^c)}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, X_{1:t}^c)}, \tag{24}$$

where the likelihood is Gaussian, $p(\mathbf{y}_t|f_t^c, X_t^c) = \mathcal{N}(H_t f_t^c, R_t)$ which can be directly obtained from Eq. (19). The prediction distribution is Gaussian,

Algorithm 1 Our KNN-KFGP Approach.

```
1: for $t = 1$ to $M$ do
2:     Applying KNN-based mechanism to find the training data collection
       $(X_t, \mathbf{y}_t) = \{(\mathbf{x}_i, y_i)\}_{i=1}^{K_t}$ for $\mathbf{x}_\star^t$
3:     Computing $G_t$, $Q_t$, $R_t$ in Eq. (16), (17), (19) for state space model
4:     Performing the *predict* step of Kalman filter: Eq. (22)-(23)
5:     Performing the *update* step of Kalman filter: Eq. (26)-(28)
6: end for
```

$p(f_t^c|\mathbf{y}_{1:t-1}, X_{1:t}^c) = \mathcal{N}(\mu_{t|t-1}^c, \Sigma_{t|t-1}^c)$ in Eq. (22)-(23). Consequentially, the marginal distribution is also Gaussian,

$$\begin{aligned} & p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, X_{1:t}^c) \\ =& \int p(\mathbf{y}_t|f_t^c, X_t^c)p(f_t^c|\mathbf{y}_{1:t-1}, X_{1:t}^c)df_t^c \\ =& \int \mathcal{N}(H_t f_t^c, R_t)\mathcal{N}(\mu_{t|t-1}^c, \Sigma_{t|t-1}^c)df_t^c \\ =& \mathcal{N}(H_t\mu_{t|t-1}^c, H_t\Sigma_{t|t-1}^c H_t^T + R_t). \end{aligned} \tag{25}$$

In this case, we put Eq. (19), (21), (25) into (24) and we can obtain that the t_{th} posterior is Gaussian, $p(f_t^c|\mathbf{y}_{1:t}, X_{1:t}^c) = \mathcal{N}(\mu_{t|t}^c, \Sigma_{t|t}^c)$ with the following updates,

$$\Gamma_t = \Sigma_{t|t-1}^c H_t^T(H_t\Sigma_{t|t-1}^c H_t^T + R_t)^{-1}, \tag{26}$$

$$\mu_{t|t}^c = \mu_{t|t-1}^c + \Gamma_t(\mathbf{y}_t - H_t\mu_{t|t-1}^c), \tag{27}$$

$$\Sigma_{t|t}^c = \Sigma_{t|t-1}^c - \Gamma_t H_t \Sigma_{t|t-1}^c, \tag{28}$$

where Γ_t is called the Kalman Gain. Finally, one can straightforwardly obtain the predictive distribution $\mathcal{N}(\mu_\star^t, \sigma_\star^t)$ over the latent function value $f(\mathbf{x}_\star^t)$, i.e., *our regression goal*, from the joint posterior $p(f_t^c|\mathbf{y}_{1:t}, X_{1:t}^c) = \mathcal{N}(\mu_{t|t}^c, \Sigma_{t|t}^c)$. Specifically, due to $f_t^c = \{f(X_t), f(\mathbf{x}_\star^t)\}$, the mean $\mu_\star^t$ and variance $\sigma_\star^t$ for $f(\mathbf{x}_\star^t)$ is directly from the corresponding element of $\mu_{t|t}^c$ and $\Sigma_{t|t}^c$.

The whole approach is summarized in Algorithm 1. In our KNN-KFGP, Kalman Filter is used as a sequential inference tool to recursively estimate the latent function values at test inputs for regression. Compared to the complexity of GP ($O(N^3)$), the main computation of our KNN-KFGP is contributed to Kalman filter $O(MK^3)$ and KNN search $O(MN)$ after various precomputations [31], where N is the total size of training set, M is the number of test inputs, K is the maximum value of K_t, and K_t is the size of training collection for the t_{th} test. Hence, our KNN-KFGP can be more computationally-efficient than GP, especially when the data set is large-scale. Moreover, GP is often infeasible to model non-stationarity. On the contrary, our KNN-KFGP allows each test point to find its strongly-correlated training subset, and it thus is a suitable way to deal with the non-stationary phenomenon. Next, it is worth mentioning the difference between KFGP in [23] and our KNN-KFGP. Due to performing Kalman filter, the computation of KFGP in [23] is mainly governed by $(O(MK^3)$ that is similar to our KNN-KFGP. However, the blindly training data selection for each test in KFGP [23] often deteriorates the estimation performance, especially when the data set exhibits non-stationarity. In contrast, KNN-based mechanism of our KNN-KFGP can discover the local data properties for each test and consequentially improve the inference accuracy.

Finally, to avoid confusion, we further clarify the difference between our approach and the works [36, 37] in target tracking domain. *First of all*, our goal is regression. we propose to establish a sequential model (i.e., state space model) for GP regression, in order to model large-scale data sets recursively and thus reduce the computational burden in regression. Hence, the task in our paper is different from the one in target tracking domain. *Furthermore*, because of the difference in the task, the meaning of the states in our paper is different from the target tracking domain. In our KNN-KFGP approach, the states at t are the latent function values $f_t^c = f(X_t^c) = \{f(X_t), f(\mathbf{x}_\star^t)\}$, since our problem is regression. On the contrary, the states at t in the target tracking systems are position, velocity and acceleration of a moving object.

In the next section, we perform our KNN-KFGP on a number of synthetic and real-world data sets to show that our KNN-KFGP can achieve a superior performance, compared to the related GP regression approaches.

4. Experiments

In this section, we evaluate our KNN-KFGP on several data sets to show its validity for regression. We aim to achieve two objectives by using these experiments. *Firstly*, if the data sets can be successfully modeled by the stationary GP, we evaluate whether our KNN-KFGP is a computationally-efficient and accurate approximation of the stationary GP. The pendulum and kin-40k data sets[2] are used to validate this objective. *Secondly*, if the data sets cannot be suitably modeled by the stationary GP, we test whether our KNN-KFGP is able to deal with non-stationarity. The mobile robot perception simulation and global land-surface precipitation data sets[3] are chosen for this objective.

As mentioned, we focus on regression. For each data set, we are given a training set and a test set. The input-output data pairs in the training set are used to train our KNN-KFGP model. The input-output data pairs in the test set are used to evaluate the prediction performance. Specifically, the predictive accuracy is evaluated by the test Standardized Mean Square Error (SMSE)

$$\frac{1}{M}\sum_{t=1}^{M}\frac{(y_\star^t-\mu_\star^t)^2}{variance(\mathbf{y}_\star)},$$

and the test Mean Negative Log Probability (MNLP)

$$\frac{1}{M}\sum_{t=1}^{M}\left(\frac{(y_\star^t-\mu_\star^t)^2}{(\sigma_\star^t)^2+\sigma^2}+\log[(\sigma_\star^t)^2+\sigma^2]+\log 2\pi\right),$$

where the predictive mean $\mu_\star^t$ and variance $\sigma_\star^t$ for the latent value $f(\mathbf{x}_\star^t)$ in the test set are from Eq. (27) and (28), σ^2 is the learned noise variance, $\mathbf{y}_\star = \{y_\star^t\}_{t=1}^M$ are the ground truth test outputs which are only used for accuracy evaluation (Note that, they are not used in the training procedure). Finally, the computational efficiency is evaluated by Running Time (RT).

4.1. The Pendulum Data

The first data set is a realistic simulation of a mechanical pendulum. There are 315 training and 315 test input-output pairs in this data set. Both the training and test sets are randomly-ordered. In each pair, the input is a 9-dimensional pendulum parameter vector, and its corresponding output is an angular velocity of the pendulum.

In this experiment, we compare our KNN-KFGP to GP and its efficient variants including Kalman Filter Gaussian Process (KFGP) [23], Sparse

[2]The pendulum and kin-40k data set are available at: http://www.tsc.uc3m.es/ miguel
[3]The precipitation data set is available at: http://www2.cgd.ucar.edu

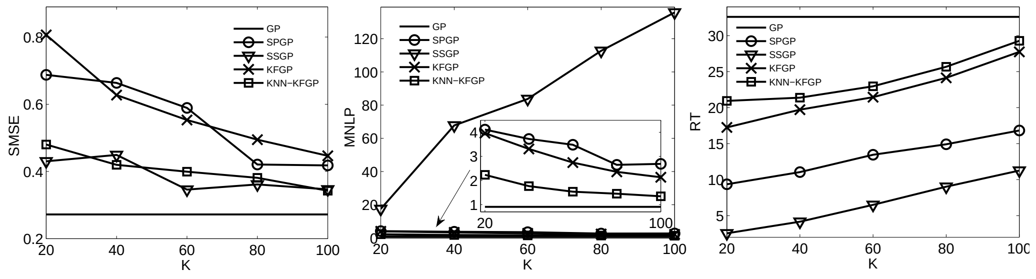


Figure 2: Prediction Performance for Pendulum Data

Pseudo-input Gaussian Process (SPGP) [19], Sparse Spectrum Gaussian Process (SSGP) [20]. We report the prediction performance as a function of K in Figure 2, where K respectively represents the size of training collection for each test point in KFGP and KNN-KFGP, the size of pseudo-input set for each test point in SPGP, the number of the basis function pairs for each test point in SSGP. Moreover, we learn the model parameters using gradient optimization with the full training set. For each K, we run all the methods three times and calculate the average SMSE, MNLP and RT (seconds).

Since the prediction for each test point in the stationary GP is made with the whole training set, we use this result as the base line. We find that the stationary GP successfully captures data flexibility with the best SMSE and MNLP. However, the tradeoff is the highest computation load. On the contrary, SSGP is more computationally efficient than stationary GP. The SMSE of SSGP is also good when K is small. But its MNLP becomes poorly deteriorated as K increases. The reason is that SSGP falls into the overfitting trap [20]. Compared to SSGP, SPGP is more acceptable for this data set. Additionally, the accuracy of KFGP is slightly better than SPGP, but the computation load is higher than SPGP.

Hence, there is a tradeoff between computation load and accuracy for all the methods above. It is worth mentioning that the pendulum data is small, so all the methods are competitively fast. However, in terms of accuracy, our KNN-KFGP is the most suitable choice among all the approximate GP approaches. In the following, we further evaluate all the methods on a large data set to show that our KNN-KFGP outperforms other methods in terms of tradeoff between accuracy and computational efficiency.

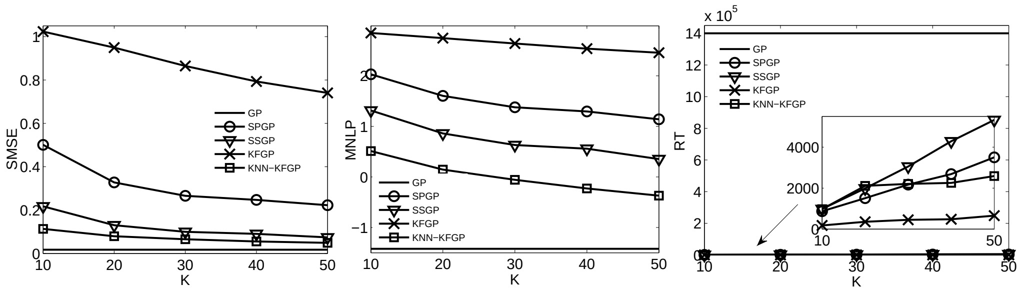


Figure 3: Prediction Performance for Kin-40k Data

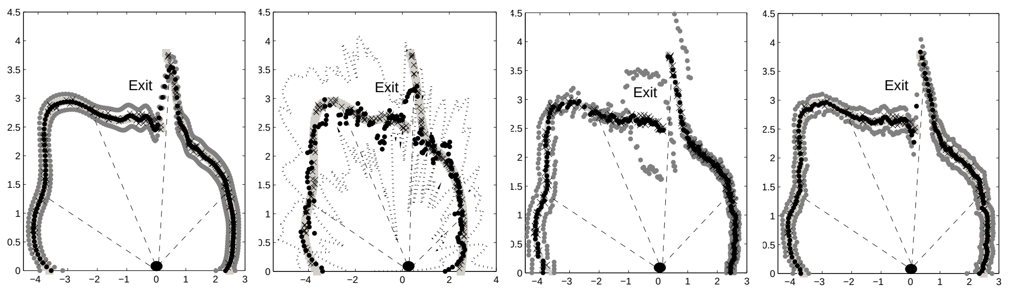
4.2. The Kin-40k Data

The second data set is a realistic simulation of a robot arm. The randomly-ordered training (10000 instances) and test (30000 instances) set are massive. In each instance, the input is a 8-dimension parameter vector and the output is scalar. In Figure 3, we show the prediction performance for all the methods (stationary GP, SPGP, SSGP, KFGP and our KNN-KFGP) as a function of K. The notation of K is the same as the pendulum experiment. Because this data set is massive, we learn the model parameters using gradient optimization with a randomly selected training subset including 1000 points. For each K, we run all the methods three times and calculate the average SMSE, MNLP and RT(seconds).

The stationary GP makes prediction for each test point with the whole training set, so its accuracy (SMSE and MNLP) is the best among these approaches. However, the computation burden is unsatisfactorily high. In contrast, KFGP is computationally-efficient while its accuracy is poor. Furthermore, the computation load of SPGP, SSGP, our KNN-KFGP is similar and much better than the stationary GP. But the best accuracy among all the approximation methods is obtained by our KNN-KFGP. Hence, compared to other approximate GPs, our KNN-KFGP is a more accurate and efficient approach for large data sets.

4.3. Mobile Robot Perception

Environment perception is a fundamental task for mobile robot systems [16]. For a mobile robot, it is essential to correctly interpret the sensor information in the non-stationary environment to find its precise location. Here we consider a perception task to evaluate how well our KNN-KFGP can deal with non-stationarity. Concretely, a mobile robot is assumed to be located at



(a) GP (b) KFGP (c) LGP (d) KNN-KFGP

Figure 4: Mobile Robot Perception. A mobile robot (black ellipse at (0,0)) uses its range sensors (four black dashed lines) to obtain distance measurements (black crosses) for obstacle (grey squares) detection. The prediction mean for all the methods are represented by black dots. The 95% confidence interval for KFGP are represented by grey dotted line, for other methods it is represented by grey dots.

$(0, 0)$ with $\pi/2$ heading. To find an exit, it uses its range sensors for obstacle detection. We simulate 200 training points where the input is a bearing angle in $[0, \pi]$ and the output is the corresponding distance measurement. The test inputs are from 0 to π with an interval $\pi/180$. The observations at different sides of the exit show the discontinuity of the measured data, which is a case of spatial non-stationarity.

In this experiment, we compare our KNN-KFGP to the stationary GP, KFGP and local GP (LGP) [31] which is a related work using local GP experts to capture non-stationarity. The notation of K in KFGP and our KNN-KFGP is the same as the first two experiments. In LGP, K is the number of local GP experts. We learn the model parameters using gradient optimization with the full training set, then run all the methods three times from $K = 1$ to 5. We find that our KNN-KFGP shows the best performance when $K = 2$, KFGP and LGP represents the best performance when $K = 5$. It indicates that the accuracy of our KNN-KFGP is not monotonically increasing in the non-stationary cases when K is increasing. The underlying reason is that, if the K is too large, some disturbed or even wrongly-correlated training points will be selected for each test point. That is also why the stationary GP cannot correctly model non-stationarity. Hence, in our experiment, we choose the best K for all the related approaches.

The results of all the methods are illustrated in Figure 4 and Table 1. In Figure 4(a), the standard GP mistakenly predicts that there is no exit, even though the robot has detected the data on both sides of the exit. In Figure

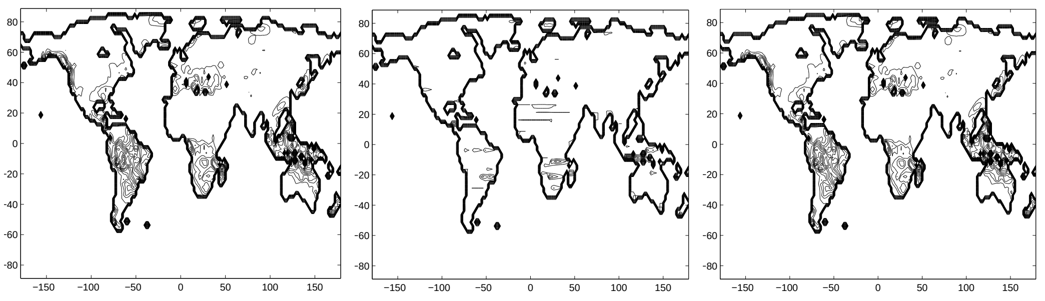
4(b), KFGP fails to make prediction since the randomly-selected training points in this case are disturbed. In Figure 4(c), LGP correctly finds the exit, but the uncertainty at the exit is high. In Figure 4(d), our KNN-KFGP successfully models the exit since the training subset for each test is strongly-correlated to that test. Finally, Table 1 also shows that the random training data selection in KFGP leads to its low accuracy. In order to improve the accuracy, KFGP has to use more training data for each test. In this case, its performance is prone to the stationary GP which is also poor for non-stationarity. LGP and our KNN-KFGP are suitable approaches for non-stationarity modeling, but our KNN-KFGP has a higher accuracy than LGP.

4.4. Global Land-Surface Precipitation

In practice, the environmental data sets usually exhibit non-stationarity [8]. Hence, we choose a global land-surface precipitation (January 2010) data set to evaluate if our KNN-KFGP can handle non-stationarity. There are 3306 points in the training set. For each point, the input is the location and the output is the precipitation value. The test set is based on a regular grid with a spatial resolution of $2.5^o \times 2.5^o$ latitude by longitude. As before, we compare our KNN-KFGP to the stationary GP, KFGP and LGP. The notation of K is the same as the robot perception experiment. We train the model parameters using gradient optimization with a randomly selected training subset including 1000 points, then run all the methods three times from $K = 1$ to 10. Our KNN-KFGP shows the best performance when $K = 5$, KFGP and LGP represents the best performance when $K = 10$. For all the methods, we show the results with the best K. In Figure 5, the prediction of KFGP almost fails due to the random training set selection. In contrast, our KNN-KFGP flexibly learns the land-surface precipitation model. The underlying reason is that each test point is learned with its strongly-correlated training set and the predict-update KF mechanism preserves the accuracy. From Table 2, it is also shown that LGP and our KNN-KFGP are suitable approaches for this non-stationarity, but our KNN-KFGP has a better accuracy than LGP. Finally, it is worth mentioning that RT of our KNN-KFGP is 475s which is much more efficient than GP (15374s).

5. Conclusion

In this paper, we propose a novel KNN-based Kalman filter for GP regression (KNN-KFGP). We firstly design a test-input-driven KNN mechanism to



(a) Observation (b) KFGP (c) KNN-KFGP

Figure 5: Global Land-Surface Precipitation. The contour in all three subplots represents the level of precipitation value. When the contours are concentrated, the precipitation value is high. Otherwise, the precipitation value is low. The subplots of KFGP and KNN-KFGP show the predicted mean of the global land-surface precipitation.

Accuracy Evaluation	SMSE	MNLP
GP	0.0130	-2.1775
LGP	0.0093	-2.3216
KFGP	0.2218	0.0791
KNN-KFGP	0.0057	-2.5060

Table 1: Accuracy Evaluation for Robot Perception Data.

construct a strongly-correlated training collection for each test. Then we take advantage of GP to establish a novel state space model, where the states are the latent function values of KNN-based data collections. Next, we propose to perform Kalman filter to infer the latent function values in a predict-update manner. As a result, Our KNN-KFGP can preserve computation efficiency due to the recursive Bayesian inference framework of Kalman filter. Meanwhile, our KNN-KFGP can capture non-stationarity, as our KNN mechanism provides each test with its strongly-correlated training set. Our experimental results show that our KNN-KFGP often achieves a superior performance in terms of accuracy and efficiency, compared with a number of related works. In the future, it would be interesting to explore an adaptive manner to learn K for each test to further improve model flexibility. We can also explore how to extend our approach with incremental local Gaussian Regression [38] in combination with an efficient sparsification [39].

Accuracy Evaluation	SMSE	MNLP
GP	0.0128	7.6863
LGP	0.0036	7.5979
KFGP	1.2582	12.1196
KNN-KFGP	0.0027	7.1255

Table 2: Accuracy Evaluation for Land Precipitation Data.

Acknowledgments

This work has been supported by Natural Sciences and Engineering Research Council of Canada (NSERC), National Natural Science Foundation of China (61502470), Shenzhen Research Program (JCYJ20160229193541167).

Reference

[1] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[2] D. Barber, Bayesian Reasoning and Machine Learning, Cambridge University Press, 2012.

[3] C. E. Rasmussen, C. K. I. Williams, Gaussian Process for Machine Learning, MIT Press, 2006.

[4] C. Plagemann, D. Fox, W. Burgard, Efficient failure detection on mobile robots using particle filters with gaussian process proposals., in: International Joint Conference on Artificial Intelligence (IJCAI'07), Conference on, 2007, pp. 2185–2190.

[5] J. Ko, D. Fox, Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models, Vol. 27, Springer, 2009, pp. 75–90.

[6] M. P. Deisenroth, C. E. Rasmussen, D. Fox, Learning to control a low-cost manipulator using data-efficient reinforcement learning, in: Robotics: Science and Systems (RSS'11), Conference on, 2011.

[7] S. Anderson, T. D. Barfoot, C. H. Tong, S. Särkkä, Batch nonlinear continuous-time trajectory estimation as exactly sparse gaussian process regression, Autonomous Robots 39 (3) (2015) 221–238.

[8] C. Paciorek, M. Schervish, Nonstationary covariance functions for gaussian process regression, Vol. 16, 2004, pp. 273–280.

[9] K. L. Silversides, A. Melkumyan, D. Wyman, Fusing gaussian processes and dynamic time warping for improved natural gamma signal classification, Mathematical Geosciences 48 (2) (2016) 187–210.

[10] M. Lázaro-Gredilla, M. K. Titsias, Variational heteroscedastic gaussian process regression, in: International Conference on Machine Learning (ICML'11), Conference on, 2011.

[11] J. Han, X.-P. Zhang, Financial time series volatility analysis using gaussian process state-space models, in: IEEE Global Conference on Signal and Information Processing (GlobalSIP.15), IEEE, 2015, pp. 358–362.

[12] J. Quinonero-Candela, C. E. Rasmussen, A Unifying View of Sparse Approximate Gaussian Process Regression, Journal of Machine Learning Research 6 (2005) 1939–1959.

[13] K. Chalupka, C. K. I. Williams, I. Murray, A Framework for Evaluating Approximation Methods for Gaussian Process Regression, Journal of Machine Learning Research 14 (2013) 333–350.

[14] Y. Gal, M. van der Wilk, C. Rasmussen, Distributed variational inference in sparse gaussian process regression and latent variable models, in: Neural Information Processing Systems (NIPS'14) Conference on, 2014, pp. 3257–3265.

[15] M. Kuss, Gaussian process models for robust regression, classification, and reinforcement learning, Ph.D. thesis, Technische Universität Darmstadt (2006).

[16] C. Plagemann, Gaussian processes for flexible robot learning, Ph.D. thesis, Albert-Ludwigs-Universität Freiburg (2008).

[17] Y. Wang, Interactions Between Gaussian Processes and Bayesian Estimation, Ph.D. thesis, Computer Science and Software Engineering Dept, Laval University, Québec (2014).

[18] K. H. Low, J. Chen, T. N. Hoang, N. Xu, P. Jaillet, Recent advances in scaling up gaussian process predictive models for large spatiotemporal data, in: Dynamic Data-Driven Environmental Systems Science, Springer, 2015, pp. 167–181.

[19] E. Snelson, Z. Ghahramani, Sparse gaussian processes using pseudo-inputs, in: Neural Information Processing Systems (NIPS'05) Conference on, 2005, pp. 1257–1264.

[20] M. Lázaro-Gredilla, J. Quinonero-Candela, C. E. Rasmussen, A. R. Figueiras-Vidal, Sparse spectrum gaussian process regression, Journal of Machine Learning Research 11 (2010) 1865–1881.

[21] L. Csató, M. Opper, Sparse Online Gaussian Processes, Neural Computation 14 (3) (2002) 641–668.

[22] S. V. Vaerenbergh, M. Lázaro-Gredilla, I. Santamaría, Kernel Recursive Least-Squares Tracker for Time-Varying Regression, IEEE Transactions on Neural Networks and Learning Systems 23 (8) (2012) 1313–1326.

[23] S. Reece, S. Roberts, An introduction to gaussian processes for the kalman filter expert, in: Information Fusion (FUSION'10), 2010 13th Conference on, IEEE, 2010, pp. 1–9.

[24] C. E. Rasmussen, Z. Ghahramani, Infinite mixture of gaussian process experts, in: Neural Information Processing Systems (NIPS'02), Conference on, 2002.

[25] R. P. Adams, O. Stegle, Gaussian Process Product Models for Nonparametric Nonstationarity, in: International Conference on Machine Learning (ICML'08), Conference on, 2008, pp. 1–8.

[26] S. Reece, R. Garnett, M. Osborne, S. Roberts, Anomaly detection and removal using non-stationary gaussian processes, arXiv preprint arXiv:1507.00566.

[27] M. Lorenzi, G. Ziegler, D. C. Alexander, S. Ourselin, Efficient gaussian process-based modelling and prediction of image time series, in: Information Processing in Medical Imaging, Springer, 2015, pp. 626–637.

[28] E. Snelson, C. E. Rasmussen, Z. Ghahramani, Warped Gaussian Processes, in: Neural Information Processing Systems (NIPS'03), Conference on, 2003, pp. 337–344.

[29] M. Lázaro-Gredilla, Bayesian Warped Gaussian Processes, in: Neural Information Processing Systems (NIPS'12), Conference on, 2012, pp. 1619–1627.

[30] D. Nguyen-Tuong, J. Peters, M. Seeger, Local Gaussian Process Regression for Real Time Online Model Learning and Control, in: Neural Information Processing Systems (NIPS'08), Conference on, 2008, pp. 1193–1200.

[31] R. Urtasun, T. Darrell, Sparse probabilistic regression for activity-independent human pose inference, in: Computer Vision and Pattern Recognition (CVPR'08), IEEE Conference on, IEEE, 2008, pp. 1–8.

[32] M. Hou, Y. Wang, B. Chaib-draa, Online local gaussian process for tensor-variate regression: Application to fast reconstruction of limb movements from brain signal, in: Acoustics, Speech and Signal Processing (ICASSP'15), IEEE International Conference on, IEEE, 2015, pp. 5490–5494.

[33] D. J. C. MacKay, Introduction to Gaussian Processes, in: Neural Networks and Machine Learning, 1998, pp. 133–165.

[34] A. Doucet, N. de Freitas, N. Gordon, Sequential Monte Carlo Methods in Practice, Springer, 2001.

[35] R. E. Kalman, A new approach to linear filtering and prediction problems, Transactions of the ASME-Journal of basic Engineering 82(Series D) (1960) 35–45.

[36] X. R. Li, V. P. Jilkov, Survey of Maneuvering Target Tracking. Part I: Dynamic Models, IEEE Transactions on Aerospace and Electronic Systems 39 (2003) 1333–1364.

[37] X. Jin, X. Lian, T. Su, Y. Shi, B. Miao, Closed-Loop Estimation for Randomly Sampled Measurements in Target Tracking System, Mathematical Problems in Engineering (2014) 1–12.

[38] F. Meier, P. Hennig, S. Schaal, Incremental local gaussian regression, in: Neural Information Processing Systems (NIPS'14) Conference on, 2014, pp. 972–980.

[39] J. Schreiter, D. Nguyen-Tuong, M. Toussaint, Efficient sparsification for gaussian process regression, Neurocomputing 192 (2016) 29–37.