論文 / 著書情報
Article / Book Information

Title	Parametric Speech Synthesis Using Local and Global Sparse Gaussian Processes
Author	Tomoki Koriyama, Takashi Nose, Takao Kobayashi
Journal/Book name	Proc. The 24th IEEE International Workshop on Machine Learning for Signal Processing, Vol. , No. , pp.
Issue date	2014, 9
DOI	http://dx.doi.org/10.1109/MLSP.2014.6958921
URL	http://www.ieee.org/index.html
Copyright	(c)2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works.
Note	このファイルは著者（最終）版です。 This file is author (final) version.

2014 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2014, REIMS, FRANCE

PARAMETRIC SPEECH SYNTHESIS USING LOCAL AND GLOBAL SPARSE GAUSSIAN PROCESSES

Tomoki Koriyama[1]*, Takashi Nose*[2]*, Takao Kobayashi*[1]

[1]Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology, Japan
[2]Graduate School of Engineering, Tohoku University, Japan
koriyama@ip.titech.ac.jp, tnose@m.tohoku.ac.jp, takao.kobayashi@ip.titech.ac.jp

ABSTRACT

This paper describes an application of Gaussian process regression (GPR) to parametric speech synthesis. GPR enables us to predict synthetic speech parameters by utilizing exemplars of training speech data directly without converting the acoustic features of training data into too small number of model parameters thanks to nonparametric Bayesian regression. However, GPR inherently requires high computational cost and resources. In this paper, to alleviate this problem, we incorporate local and global sparse Gaussian process approximation into the statistical speech synthesis framework, and investigate trade-off between computational cost and speech synthesis performance through experiments. Moreover, we examine the way of choosing pseudo data set used for the sparse GP approximation.

Index Terms— parametric speech synthesis, Gaussian process regression, partially independent conditional (PIC) approximation

1. INTRODUCTION

Recently, Gaussian processes (GPs) have drawn researchers' attention because of their effectiveness and flexibility in non-parametric regression modeling of complex systems. Indeed Gaussian process regression (GPR) has been applied to various speech processing areas so far [1–4]. We have also proposed a framework of GPR-based spectral feature modeling for statistical parametric speech synthesis [5, 6].

Speech synthesis is considered to be a regression problem in which acoustic features for waveform generation are estimated from a given set of acoustic features of training data. In our proposed framework, we formulated this problem as a frame-level regression of acoustic features by defining the covariance function of frame-level contextual features. Since GPR inherently requires high computational costs and resources, it is essential to use some approximations that lead to GPR with a feasible computational cost on large data sets [7–9]. For this purpose, we employed a partially independent conditional (PIC) approximation [5]. PIC approximation [8] uses clusters to perform local GPR and pseudo data set to perform global information approximation. This results in proper modeling of not only local acoustic feature variation within a phone but also smoothly changing features among adjoining phones.

In our previous study [5, 6], we showed that the GPR-based approach outperformed the conventional hidden Markov model (HMM)-based speech synthesis. However, the evaluation was done for only one speaker, and the size and choice of pseudo data set were not investigated substantially. In this study, therefore, we evaluate the performance and computational cost of the proposed technique using four speakers' speech data and various sizes of pseudo data sets. Furthermore, we examine the way of choosing pseudo data set. Specifically, we incorporate *k-medoids* [10] clustering in order to make a well-balanced pseudo data set and compare the performance with a random choice approach.

The remainder of this paper is organized as follows. Section 2 describes a framework of GPR-based speech synthesis using the PIC approximation for speech synthesis systems and discuss the computational complexity for training and synthesis processes. Section 3 discusses the methods of choosing the pseudo data set. Section 4 shows experimental results including actual computation time and model footprint. Finally, we give our conclusions and future work in Section 5.

2. GAUSSIAN PROCESS REGRESSION FOR SPEECH SYNTHESIS

Statistical parametric speech synthesis is a technique of generating speech parameters from contextual information using statistical model. The speech parameters consist of spectral and pitch features of a short segment called a frame. Context is the phonetic and prosodic information such as how phonemes appear in an utterance and which syllables are stressed. The most successful and widely studied approach is the statistical speech synthesis based on hidden Markov models (HMMs) [11, 12], which models acoustic features corresponding to phone-level contexts by HMMs and generate speech parameters from the trained context-dependent HMMs.

In contrast, a framework we proposed in [5, 6] is based on Gaussian process regression (GPR) [13] where speech parameters are predicted from frame-level contextual features. Here let $\mathbf{x}_n$ and $y_n^{(d)}$ denote a frame-level contextual information of n-th frame and corresponding zero-mean-normalized d-th dimension's feature of an acoustic feature vector, respectively. We define a latent function $f^{(d)}(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, k(\mathbf{x}_m, \mathbf{x}_n; \boldsymbol{\theta}^{(d)}))$ where $k(\mathbf{x}_m, \mathbf{x}_n; \boldsymbol{\theta}^{(d)})$ is a kernel (covariance) function and $\boldsymbol{\theta}^{(d)}$ represents its parameter set. Using a Gaussian noise $\epsilon^{(d)} \sim \mathcal{N}(0, (\sigma_\nu^d)^2)$, the acoustic feature is represented by $y_n^{(d)} = f^{(d)}(\mathbf{x}_n) + \epsilon^{(d)}$. Suppose that a training data set is given by

$$\mathcal{D} = \{(\mathbf{x}_n, y_n^{(1)}, \ldots, y_n^{(D)}) | n = 1, \ldots, N\} \quad (1)$$

where D is a dimension of the acoustic feature vector. Similarly, a

A part of this work was supported by JSPS Grant-in-Aid for Scientific Research 24300071, 25540065, and 25·8776.

test (synthetic) data set is given by

$$\mathcal{D}_T = \{(\mathbf{x}_t, y_t^{(1)}, \ldots, y_t^{(D)}) | t = 1, \ldots, T\}. \quad (2)$$

Let $\mathbf{X}_N = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$, $\mathbf{y}_N^{(d)} = [y_1^{(d)}, \ldots, y_N^{(d)}]^\top$, and $\mathbf{Y}_N = [\mathbf{y}_N^{(1)}, \ldots, \mathbf{y}_N^{(D)}]$ be matrix forms of all input and output variables of training data, and $\mathbf{f}_N^{(d)} = [f^{(d)}(\mathbf{x}_1), \ldots, f^{(d)}(\mathbf{x}_N)]^\top$ and $\mathbf{F}_N = [\mathbf{f}_N^{(1)}, \ldots, \mathbf{f}_N^{(D)}]$ be the latent function values of the training data. Moreover, $\mathbf{X}_T$, $\mathbf{y}_T$, $\mathbf{Y}_T$, $\mathbf{f}_T$, and $\mathbf{F}_T$ denote matrix forms for test data in the same way as the training data. Here we assume that each dimension is independent and kernel function parameters $\boldsymbol{\theta}^{(d)}$ and noise variances $(\sigma_\nu^{(d)})^2$ are the same among all dimensions. Then let $\boldsymbol{\theta}$ and σ_ν^2 be the common hyperparameter vector and noise variance, respectively. Under these notations and assumptions, the predictive distribution of synthetic speech parameters is given by a matrix variate normal distribution [14] as follows:

$$\begin{aligned} p(\mathbf{Y}_T | \mathbf{Y}_N, \boldsymbol{\theta}) &= \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_T^{(d)}; \boldsymbol{\mu}_{\mathbf{y}_T|\mathbf{y}_N}^{(d)}, \boldsymbol{\Sigma}_{\mathbf{y}_T|\mathbf{y}_N}^{(d)}) \\ &= \mathcal{MN}_{T,D}(\mathbf{Y}_T; \mathbf{M}_{\mathbf{y}_T|\mathbf{y}_N}, \boldsymbol{\Sigma}_{\mathbf{y}_T|\mathbf{y}_N}, \mathbf{I}_D) \quad (3) \end{aligned}$$

where $\mathbf{M}_{\mathbf{y}_T|\mathbf{y}_N} = [\boldsymbol{\mu}_{\mathbf{y}_T|\mathbf{y}_N}^{(1)}, \ldots, \boldsymbol{\mu}_{\mathbf{y}_T|\mathbf{y}_N}^{(D)}]$ and a matrix variate normal distribution is defined by

$$\begin{aligned} \mathcal{MN}_{N,D}(\mathbf{Y}; \mathbf{M}, \boldsymbol{\Sigma}, \mathbf{V}) &\stackrel{\text{def}}{=} \frac{1}{(2\pi)^{ND/2} |\mathbf{V}|^{D/2} |\boldsymbol{\Sigma}|^{N/2}} \\ &\cdot \exp\left(-\frac{1}{2} \mathrm{Tr}\left[\mathbf{V}^{-1} (\mathbf{Y} - \mathbf{M})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{Y} - \mathbf{M}) \right] \right) \quad (4) \end{aligned}$$

when $\mathbf{Y}$, $\mathbf{M}$, $\boldsymbol{\Sigma}$, and $\mathbf{V}$ are $N \times D$, $N \times D$, $N \times N$, and $D \times D$ matrices, respectively. $\mathbf{M}_{\mathbf{y}_T|\mathbf{y}_N}$ and $\boldsymbol{\Sigma}_{\mathbf{y}_T|\mathbf{y}_N}$ correspond to the predictive mean and covariance matrices and are obtained by

$$\mathbf{M}_{\mathbf{y}_T|\mathbf{y}_N} = \mathbf{K}_{TN} [\mathbf{K}_N + \sigma_\nu^2 \mathbf{I}_N]^{-1} \mathbf{Y}_N \quad (5)$$

$$\boldsymbol{\Sigma}_{\mathbf{y}_T|\mathbf{y}_N} = \mathbf{K}_T - \mathbf{K}_{TN} [\mathbf{K}_N + \sigma_\nu^2 \mathbf{I}_N]^{-1} \mathbf{K}_{NT} \quad (6)$$

where $\mathbf{K}_N$, $\mathbf{K}_T$, and $\mathbf{K}_{NT} = \mathbf{K}_{TN}^\top$ are kernel matrices given by

$$\begin{aligned} &\mathbf{K}_N = K(\mathbf{X}_N, \mathbf{X}_N), \quad \mathbf{K}_T = K(\mathbf{X}_T, \mathbf{X}_T) \\ &\mathbf{K}_{NT} = K(\mathbf{X}_N, \mathbf{X}_T). \end{aligned} \quad (7)$$

$K(\mathbf{X}_1, \mathbf{X}_2)$ denotes a matrix whose (i, j)-th element is evaluated by a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the i-th and j-th row vectors in $\mathbf{X}_1$ and $\mathbf{X}_2$, respectively.

2.1. Kernel definition

The choice of kernel function is crucial for obtaining better performance in speech synthesis. Generally, it has been shown that the kernel function that expresses similarity works well. Here we consider following principles concerning frames:

- Nearby frames have high correlation.
- Similar phonemes have high correlation.

On the basis of these principles, we define a kernel function as follows [5]:

$$\begin{aligned} &k(\mathbf{x}_m, \mathbf{x}_n; \boldsymbol{\theta}) = \\ &\sum_{i \in \{-1,0,+1\}} \sum_{j \in \{-1,0,+1\}} w_m^{(i)} w_n^{(i)} k_p(\mathbf{p}_m^{(i)}, \mathbf{p}_n^{(j)}) k_c(\mathbf{c}_m^{(i)}, \mathbf{c}_n^{(j)}) \\ &+ \delta_{mn} \theta_{\text{floor}}^2 \end{aligned} \quad (8)$$

where the last term $\delta_{mn}\theta_{\text{floor}}^2$ is a flooring value to keep kernel matrices positive definite. Two kernel functions $k_p(\cdot)$ and $k_c(\cdot)$, which correspond to the position kernel and the phone context kernel, respectively, are introduced in order to satisfy the principles. Specifically, the position kernel represents the position similarity in phones and the phone context kernel expresses that of phonetic features such as articulation manner and place. In this study, we use the sum of squared exponential (SE) kernel for the position kernel and linear kernel for the phone context kernel. The sum operation in (8) is employed to keep value changes smooth.

The input feature of kernel function is defined as a set of position and phone contexts of adjacent phones:

$$\mathbf{x}_n = (\mathbf{w}_n, \mathbf{P}_n, \mathbf{C}_n) \quad (9)$$

where $\mathbf{w}$, $\mathbf{p}$, and $\mathbf{C}$ are sets of weights, position contexts, and phone contexts expressed as

$$\mathbf{w}_n = \{w_n^{(-1)}, w_n^{(0)}, w_n^{(+1)}\} \quad (10)$$

$$\mathbf{P}_n = \{\mathbf{p}_n^{(-1)}, \mathbf{p}_n^{(0)}, \mathbf{p}_n^{(+1)}\} \quad (11)$$

$$\mathbf{C}_n = \{\mathbf{c}_n^{(-1)}, \mathbf{c}_n^{(0)}, \mathbf{c}_n^{(+1)}\}. \quad (12)$$

The superscripts -1, 0, and $+1$ of the variables correspond to the preceding, current, and succeeding phones of the current frame. $\mathbf{p}_n^{(j)}$ is a position context that includes the relative positions such as phone-normalized position and positions from the beginning and end of the phone. $\mathbf{c}_n^{(j)}$ is a binary-valued distinctive phonetic feature (DPF) [15] vector including preceding, current, and succeeding phonemes. A weight parameter $w_n^{(i)}$ is used to emphasize the effect of closer phones to the n-th frame.

2.2. Sparse GP approximation for feasible computation

One of the issues of applying GPR to a practical system is computational feasibility. For example, matrix inversion in (5) and (6) requires $\mathcal{O}(N^3)$ computations and $\mathcal{O}(N^2)$ memory footprint if the number of frames in the training data is N. Furthermore, when synthesizing speech, we need to pick up all frames in the training data to compute the kernel matrix $\mathbf{K}_{TN}$. This is intractable and unrealistic as a practical speech synthesis system.

Sparse approximation is an effective way of reducing computational cost, which assumes the sparsity of kernel function and reduces computational cost by calculating sub-matrices of the kernel matrix. One approach to the sparse approximation is the use of local GPs [8] by dividing all data into multiple blocks and compute kernel matrices only inside of the blocks. However, this approach fails to discontinuity at the boundaries of local blocks. Therefore our approach chooses a partially independent conditional (PIC) approximation [8], which is also called local and global sparse GPs. In the PIC approximation, the kernel function value between training data point and the point of pseudo data set, called inducing input, is calculated and the pseudo data set works as a smoother at the block boundaries.

We need to specify the pseudo data set and blocks to use PIC approximation. The choice of pseudo data set will be explained in Sect. 3. As for the blocks, we utilize decision-tree-based context clustering [16] that has been effectively used in the HMM-based speech synthesis. The decision tree is constructed by splitting nodes using a question about phone-level context and resultant leaf nodes have similar acoustic features. We stop splitting nodes if the node has less than a prescribed value of B, and we use the leaf nodes as the blocks.

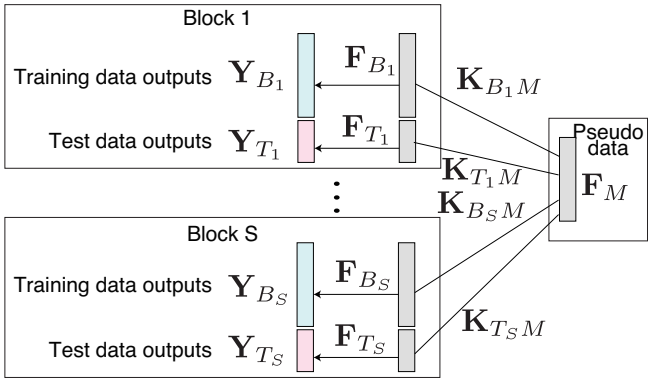

Fig. 1. Graphical representation of partially independent conditional (PIC) approximation.

Figure 1 shows a graphical representation of the PIC approximation. Let $\mathbf{F}_M$ be a latent variable of pseudo data inputs and the distribution of $\mathbf{F}_M$ be given by following matrix variate normal distribution:

$$p(\mathbf{F}_M|\boldsymbol{\theta}) = \mathcal{MN}_{M,D}(\mathbf{F}_M; \mathbf{O}_{MD}, \mathbf{K}_M, \mathbf{I}_D) \tag{13}$$

where $\mathbf{K}_M$ is a covariance (Gram) matrix of pseudo input features. $\{\mathbf{Y}_{B_s}(s = 1, \ldots, S)\}$ represents the divided output variables of training data. In PIC approximation, the joint probability of all training data $\mathbf{Y}_N$ is defined by

$$\begin{aligned} p(\mathbf{Y}_N|\boldsymbol{\theta}) &= \int p(\mathbf{Y}|\mathbf{F}_M, \boldsymbol{\theta}) p(\mathbf{F}_M|\boldsymbol{\theta}) d\mathbf{F}_M \\ &= \int \prod_{s=1}^{S} p(\mathbf{Y}_{B_s}|\mathbf{F}_M, \boldsymbol{\theta}) p(\mathbf{F}_M|\boldsymbol{\theta}) d\mathbf{F}_M \end{aligned} \tag{14}$$

where

$$p(\mathbf{Y}_{B_s}|\mathbf{F}_M, \boldsymbol{\theta}) = \mathcal{MN}_{B_s,D}(\mathbf{Y}_{B_s}; \mathbf{K}_{B_s M}\mathbf{K}_M^{-1}\mathbf{F}_M, \boldsymbol{\Lambda}_{B_s}, \mathbf{I}_D) \tag{15}$$

$$\boldsymbol{\Lambda}_{B_s} = \mathbf{K}_{B_s} - \mathbf{Q}_{B_s} + \sigma_\nu^2 \mathbf{I}_{B_s} \tag{16}$$

$$\mathbf{Q}_{B_s} = \mathbf{K}_{B_s M}\mathbf{K}_M^{-1}\mathbf{K}_{M B_s}. \tag{17}$$

By marginalizing out the latent variable $\mathbf{F}_M$ in (14), we obtain the following matrix variate normal distribution.

$$p(\mathbf{Y}_N|\boldsymbol{\theta}) = \mathcal{MN}_{N,D}(\mathbf{Y}_N; \mathbf{O}_{ND}, \mathbf{K}_N^{\mathrm{PIC}} + \sigma_\nu^2 \mathbf{I}_N, \mathbf{I}_D) \tag{18}$$

$$\begin{aligned} \mathbf{K}_N^{\mathrm{PIC}} &= \mathbf{Q}_N + \mathrm{blkdiag}(\mathbf{K}_N - \mathbf{Q}_N) \\ &= \begin{bmatrix} \mathbf{K}_{B_1} & \mathbf{Q}_{B_1 B_2} & \cdots & \mathbf{Q}_{B_1 B_S} \\ \mathbf{Q}_{B_2 B_1} & \mathbf{K}_{B_2} & & \mathbf{Q}_{B_2 B_S} \\ \vdots & & \ddots & \vdots \\ \mathbf{Q}_{B_S B_1} & \mathbf{Q}_{B_S B_2} & \cdots & \mathbf{K}_{B_S} \end{bmatrix} \end{aligned} \tag{19}$$

where $\mathbf{Q}_{B_i B_j} = \mathbf{K}_{B_i M}\mathbf{K}_M^{-1}\mathbf{K}_{M B_j}$.

The predictive distribution for synthetic speech parameters is given by

$$p(\mathbf{Y}_T|\mathbf{Y}_N, \boldsymbol{\theta}) = \mathcal{MN}_{T,D}(\mathbf{Y}_T; \mathbf{M}_{\mathbf{Y}_T|\mathbf{Y}_N}, \boldsymbol{\Sigma}_{\mathbf{Y}_T|\mathbf{Y}_N}, \mathbf{I}_D) \tag{20}$$

$$\mathbf{M}_{\mathbf{Y}_T|\mathbf{Y}_N} = \begin{bmatrix} \mathbf{M}_{\mathbf{Y}_{T_1}|\mathbf{Y}_N}^\top & \cdots & \mathbf{M}_{\mathbf{Y}_{T_S}|\mathbf{Y}}^\top \end{bmatrix}^\top \tag{21}$$

$$\boldsymbol{\Sigma}_{\mathbf{Y}_T|\mathbf{Y}_N} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{Y}_{T_1 T_1}|\mathbf{Y}_N} & \cdots & \boldsymbol{\Sigma}_{\mathbf{Y}_{T_1 T_S}|\mathbf{Y}_N} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{\mathbf{Y}_{T_S T_1}|\mathbf{Y}_N} & \cdots & \boldsymbol{\Sigma}_{\mathbf{Y}_{T_S T_S}|\mathbf{Y}_N} \end{bmatrix}. \tag{22}$$

The sub-matrices $\mathbf{M}_{\mathbf{Y}_{T_s}|\mathbf{Y}_N}$ and $\boldsymbol{\Sigma}_{\mathbf{Y}_{T_i T_j}|\mathbf{Y}_N}$ are given by

$$\mathbf{M}_{\mathbf{Y}_{T_s}|\mathbf{Y}_N} = \mathbf{K}_{T_s N}^{\mathrm{PIC}} \left[\mathbf{K}_N^{\mathrm{PIC}} + \sigma_\nu^2 \mathbf{I}_N\right]^{-1} \mathbf{Y}_N \tag{23}$$

$$\begin{aligned} \boldsymbol{\Sigma}_{\mathbf{Y}_{T_i T_j}|\mathbf{Y}_N} &= \mathbf{Q}_{T_i T_j} + \delta_{T_i T_j}(\mathbf{K}_{T_i} - \mathbf{Q}_{T_i} + \sigma_\nu^2 \mathbf{I}_{T_i}) \\ &\quad + \mathbf{K}_{N T_i}^{\mathrm{PIC}} \left[\mathbf{K}_N^{\mathrm{PIC}} + \sigma_\nu^2 \mathbf{I}_N\right]^{-1} \mathbf{K}_{N T_j}^{\mathrm{PIC}} \end{aligned} \tag{24}$$

where $\mathbf{K}_{N T_s}^{\mathrm{PIC}}$ is an approximation of $\mathbf{K}_{N T_s}$ given by

$$\begin{aligned} \mathbf{K}_{N T_s}^{\mathrm{PIC}} = \big[&\mathbf{Q}_{B_1 T_s} \ \cdots \ \mathbf{Q}_{B_{s-1} T_s} \ \mathbf{K}_{B_s T_s} \\ &\mathbf{Q}_{B_{s+1} T_s} \ \cdots \ \mathbf{Q}_{B_S T_s}\big]. \end{aligned} \tag{25}$$

Since $\mathbf{K}_N^{\mathrm{PIC}} + \sigma_\nu^2 \mathbf{I}_N$ is given by the sum of the low rank matrix $\mathbf{Q}_N$ and the block diagonal matrix, we can calculate the mean and covariance effectively using the Woodbury, Sherman & Morrison formula [17]. Practically, first we calculate following GPR parameters

$$\mathbf{B} = \mathbf{K}_M + \sum_{s=1}^{S} \mathbf{K}_{M B_s} \boldsymbol{\Lambda}_{B_s}^{-1} \mathbf{K}_{B_s M} \tag{26}$$

$$\mathbf{Z} = \mathbf{B}^{-1} \sum_{s=1}^{S} \mathbf{K}_{M B_s} \boldsymbol{\Lambda}_{B_s}^{-1} \mathbf{Y}_{B_s}. \tag{27}$$

Then, the parameters of predictive distribution are calculated as follows:

$$\mathbf{M}_{\mathbf{Y}_{T_s}|\mathbf{Y}_N} = (\mathbf{K}_{T_s B_s} - \mathbf{Q}_{T_s B_s}) \boldsymbol{\Lambda}_{B_s}^{-1} \mathbf{Y}_{B_s} + \boldsymbol{\Omega}_{T_s M} \mathbf{Z} \tag{28}$$

$$\begin{aligned} \boldsymbol{\Sigma}_{\mathbf{Y}_{T_i T_j}|\mathbf{Y}_N} = \delta_{T_i T_j} \Big[&\mathbf{K}_{T_i} - \mathbf{Q}_{T_i} + \sigma_\nu^2 \mathbf{I}_{T_i} \\ &- (\mathbf{K}_{T_i B_i} - \mathbf{Q}_{T_i B_i}) \boldsymbol{\Lambda}_{B_i}^{-1} (\mathbf{K}_{B_i T_i} - \mathbf{Q}_{B_i T_i}) \Big] \\ &+ \boldsymbol{\Omega}_{T_i M} \mathbf{B}^{-1} \boldsymbol{\Omega}_{T_j M}^\top \end{aligned} \tag{29}$$

where

$$\boldsymbol{\Omega}_{T_s M} = \mathbf{K}_{T_s M} - (\mathbf{K}_{T_s B_s} - \mathbf{Q}_{T_s B_s}) \boldsymbol{\Lambda}_{B_s}^{-1} \mathbf{K}_{B_s M}. \tag{30}$$

Training of GPR with PIC approximation implies computing $\mathbf{B}$ and $\mathbf{Z}$, which is independent from test data, in advance. When the block size is B, the training data size N corresponds to SB and the computational cost for training results in $\mathcal{O}(N(B + M)(B + M + D))$, whereas GP without any approximations requires $O(N^3)$ computations. In order to achieve efficient computation of the predictive distribution, we store a set of matrices $\left(\{\mathbf{K}_{B_s M}\}, \mathbf{K}_M, \{\boldsymbol{\Lambda}_{B_s}^{-1}\}, \{\boldsymbol{\Lambda}_{B_s}^{-1}\mathbf{Y}_{B_s}\}, \mathbf{B}^{-1}, \mathbf{Z}\right)$ as GPR parameters, which requires $\mathcal{O}(N(B + M + D))$ memory footprint. In this case, the computational cost for the inference of predictive distribution becomes $\mathcal{O}(T(B + M)(B + M + D))$.

2.3. Optimization of kernel hyperparameters

To estimate an appropriate hyperparameter set $\boldsymbol{\theta}$ automatically for PIC approximation, we employ an empirical Bayesian approach, which maximizes the marginal likelihood $p(\mathbf{Y}_N|\boldsymbol{\theta})$ based on generalized EM algorithm. We use the latent variable $\mathbf{F}_M$ for that of EM algorithm. Q-function is defined by

$$Q(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = \int p(\mathbf{F}_M|\mathbf{Y}_N, \boldsymbol{\theta}) \log \left(p(\mathbf{Y}_N|\mathbf{F}_M, \tilde{\boldsymbol{\theta}}) p(\mathbf{F}_M|\tilde{\boldsymbol{\theta}})\right) d\mathbf{F}_M \tag{31}$$

where $\tilde{\boldsymbol{\theta}}$ is a new hyperparameter set. Using the assumption of PIC approximation, the Q-function is decomposed into two Q-functions Q_s and Q_M as follows:

$$Q(\theta, \tilde{\theta}) = \sum_{s=1}^{S} Q_s(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) + Q_M(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \tag{32}$$

$$Q_s(\theta, \tilde{\theta}) = \int p(\mathbf{F}_M|\mathbf{Y}_N, \boldsymbol{\theta}) \log p(\mathbf{Y}_{B_s}|\mathbf{F}_M, \tilde{\boldsymbol{\theta}}) d\mathbf{F}_M \tag{33}$$

$$Q_M(\theta, \tilde{\theta}) = \int p(\mathbf{F}_M|\mathbf{Y}_N, \boldsymbol{\theta}) \log p(\mathbf{F}_M|\tilde{\boldsymbol{\theta}}) d\mathbf{F}_M. \tag{34}$$

In the E-step, we calculate the statistics of pseudo-data variables by

$$\mathbb{E}[\mathbf{F}_M|\mathbf{Y}_N, \boldsymbol{\theta}] = \mathbf{K}_M \mathbf{Z} \tag{35}$$

$$\mathbb{E}[\mathbf{F}_M \mathbf{F}_M^\top|\mathbf{Y}_N, \boldsymbol{\theta}] = \mathbf{K}_M (D\mathbf{B}^{-1} + \mathbf{Z}\mathbf{Z}^\top)\mathbf{K}_M. \tag{36}$$

We require $\mathcal{O}(N(B+M)(B+M+D))$ computations for the E-step in the same way as the training of GPR. In the M-step, we employ generalized EM algorithm because it is difficult to find the exact hyperparameter $\tilde{\boldsymbol{\theta}}^*$ that maximizes Q-function. Specifically, we increase the value of Q-function using a gradient-based method. Furthermore, we use stochastic gradient descent (SGD) algorithm because the Q-function is represented by the sum of Q-functions as

$$Q(\theta, \tilde{\theta}) = \sum_{s=1}^{S} \left(Q_s(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) + \frac{B_s}{N} Q_M(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \right). \tag{37}$$

In each step of SGD, we randomly choose a block and update hyperparameters. The i-th hyperparameter at the k-th iteration is updated as

$$\begin{aligned}
\tilde{\theta}_i^{(k+1)} &= \tilde{\theta}_i^{(k)} + \eta_i^{(k)} \left(\frac{\partial Q_s(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}^{(k)})}{\partial \tilde{\theta}_i^{(k)}} + \frac{B_s}{N} \cdot \frac{\partial Q_M(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}^{(k)})}{\partial \tilde{\theta}_i^{(k)}} \right) \\
&= \tilde{\theta}_i^{(k)} + \eta_i^{(k)} \left(\mathrm{Tr}\left[\mathbf{C}_{B_s} \frac{\partial \mathbf{K}_{B_s}}{\partial \tilde{\theta}_i^{(k)}} \right] + \mathrm{Tr}\left[\mathbf{C}_{B_s M}^\top \frac{\partial \mathbf{K}_{B_s M}}{\partial \tilde{\theta}_i^{(k)}} \right] \right. \\
&\quad \left. + \mathrm{Tr}\left[\mathbf{C}_M \frac{\partial \mathbf{K}_M}{\partial \tilde{\theta}_i^{(k)}} \right] \mathrm{Tr}\left[\mathbf{C}_{B_s} \frac{\partial \sigma_\nu^2}{\partial \tilde{\theta}_i^{(k)}} \right] \right)
\end{aligned} \tag{38}$$

where $\eta_i^{(k)}$ represents a step size of i at time k. $\mathbf{C}_{B_s}$, $\mathbf{C}_{B_s M}$, and $\mathbf{C}_M$ are matrices independent from the hyperparameter index i and given by

$$\begin{aligned}
\mathbf{C}_{B_s} = \frac{1}{2} \Big(& -D\tilde{\mathbf{\Lambda}}_{B_s}^{-1} + \tilde{\mathbf{\Lambda}}_{B_s}^{-1} \mathbf{Y}_{B_s} \mathbf{Y}_{B_s}^\top \tilde{\mathbf{\Lambda}}_{B_s}^{-1} \\
& + \tilde{\mathbf{\Lambda}}_{B_s}^{-1} \tilde{\mathbf{K}}_{B_s M} \tilde{\mathbf{K}}_M^{-1} \mathbb{E}[\mathbf{F}_M \mathbf{F}_M^\top|\mathbf{Y}_N, \boldsymbol{\theta}] \tilde{\mathbf{K}}_M^{-1} \tilde{\mathbf{K}}_{M B_s} \tilde{\mathbf{\Lambda}}_{B_s}^{-1} \\
& - \tilde{\mathbf{\Lambda}}_{B_s}^{-1} \tilde{\mathbf{K}}_{B_s M} \tilde{\mathbf{K}}_M^{-1} \mathbb{E}[\mathbf{F}_M|\mathbf{Y}_N, \boldsymbol{\theta}] \mathbf{Y}_{B_s}^\top \tilde{\mathbf{\Lambda}}_{B_s}^{-1} \\
& - \left[\tilde{\mathbf{\Lambda}}_{B_s}^{-1} \tilde{\mathbf{K}}_{B_s M} \tilde{\mathbf{K}}_M^{-1} \mathbb{E}[\mathbf{F}_M|\mathbf{Y}_N, \boldsymbol{\theta}] \mathbf{Y}_{B_s}^\top \tilde{\mathbf{\Lambda}}_{B_s}^{-1} \right]^\top \Big)
\end{aligned} \tag{39}$$

$$\mathbf{C}_{B_s M} = \tilde{\mathbf{\Lambda}}_{B_s}^{-1} \mathbf{D} - 2\mathbf{C}_{B_s} \tilde{\mathbf{K}}_{B_s M} \tilde{\mathbf{K}}_M^{-1} \tag{40}$$

$$\begin{aligned}
\mathbf{C}_M = & \tilde{\mathbf{K}}_M^{-1} \tilde{\mathbf{K}}_{M B_s} \mathbf{C}_{B_s} \tilde{\mathbf{K}}_{B_s M} \tilde{\mathbf{K}}_M^{-1} - \tilde{\mathbf{K}}_M^{-1} \tilde{\mathbf{K}}_{M B_s} \tilde{\mathbf{\Lambda}}_{B_s}^{-1} \mathbf{D} \\
& + \frac{B_s}{N} \left(\tilde{\mathbf{K}}_M^{-1} \mathbb{E}[\mathbf{F}_M \mathbf{F}_M^\top|\mathbf{Y}_N, \boldsymbol{\theta}] \tilde{\mathbf{K}}_M^{-1} - D\tilde{\mathbf{K}}_M^{-1} \right)
\end{aligned} \tag{41}$$

$$\begin{aligned}
\mathbf{D} = & \mathbf{Y}_{B_s} \mathbb{E}[\mathbf{F}_M|\mathbf{Y}_N, \boldsymbol{\theta}]^\top \tilde{\mathbf{K}}_M^{-1} \\
& - \tilde{\mathbf{K}}_{B_s M} \tilde{\mathbf{K}}_M^{-1} \mathbb{E}[\mathbf{F}_M \mathbf{F}_M^\top|\mathbf{Y}_N, \boldsymbol{\theta}] \tilde{\mathbf{K}}_M^{-1}.
\end{aligned} \tag{42}$$

Table 1. Amounts of training and test data [sec].

	FSM	FKS	MHT	MMY	Avg.
Training	2498	2316	2511	2177	2376 (39.6 min)
Test	218	199	211	178	202 (3.36 min)

Consequently, in each step of SGD, we require $\mathcal{O}((B+M)^2(B+M+D))$ computations for the matrices $\mathbf{C}_{B_s}$, $\mathbf{C}_{B_s M}$, and $\mathbf{C}_M$, and $\mathcal{O}(K(B+M)^2)$ for updating the hyperparameter vector where K is the number of hyperparameters.

3. PSEUDO DATA SELECTION

As described in Sect. 2.2, pseudo data set is utilized for avoiding the discontinuity of speech parameters at block boundaries because the correlation of the frames of different blocks is represented indirectly using the pseudo inputs. Hence, it is crucial to choose an appropriate pseudo data set in order to generate natural-sounding speech. Since pseudo inputs are hyperparameters of GPR as well as the parameters of kernel function, they can be optimized by an empirical Bayesian framework. However, when an input feature is a high dimensional vector and the pseudo data size is large, it is intractable to optimize all hyperparameters.

Therefore we consider to take simple ways of making pseudo data set. One of the ways is to choose samples randomly from training data. Although this method is simple, there is a possibility to pick up bad samples and fail to represent the frame correlation appropriately. Another way is to use a clustering algorithm to make pseudo inputs well-balanced. We employ k-medoids algorithm [10], which is available for structured data, as a clustering algorithm. In the k-medoids algorithm, a medoid of cluster c, denoted by $\mathbf{m}_c$, is chosen so that it has the smallest sum of distances to all other samples in the cluster c:

$$\mathbf{m}_c = \underset{\mathbf{x}_i : z_i = c}{\mathrm{argmin}} \sum_{\mathbf{x}_j : z_j = c} d(\mathbf{x}_i, \mathbf{x}_j) \tag{43}$$

where z_j is the cluster index of the sample $\mathbf{x}_j$, and d is the distance given by

$$d(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{x}_j, \mathbf{x}_j) - 2k(\mathbf{x}_i, \mathbf{x}_j). \tag{44}$$

Using the medoids, the cluster index z_i is updated by choosing a nearest medoid:

$$z_i = \underset{c}{\mathrm{argmin}}\, d(\mathbf{x}_i, \mathbf{m}_c). \tag{45}$$

Then, we use the resultant medoids as the pseudo data. In addition, as can be seen in (44), the distance depends on kernel function. Hence, we update the pseudo data set in every EM-step. Furthermore, as initial clusters, we use those obtained by state-level context clustering.

In general, a large number of pseudo data inputs leads to a better results, but accompanies increase of computational cost and footprint. Hence it becomes trade-off between them. We will experimentally examine such a trade-off in the next section.

4. EXPERIMENTS

4.1. Experimental conditions

We used two female (FSM, FKS) and two male (MHT, MMY) speakers' speech data. 503 phonetically balanced sentences uttered

Table 2. Mel-cepstral distances between original and generated speech as a function of pseudo data size [dB]

Pseudo data size	FSM	FKS	MHT	MMY	Avg.
128	5.15	4.68	*4.45*	5.23	4.88
256	5.16	4.59	4.42	5.25	4.86
512	*5.30*	4.57	4.37	5.20	4.86
1024	5.05	4.50	4.34	5.07	4.74
2048	**5.03**	**4.46**	**4.23**	**5.06**	**4.70**
HMM	*5.41*	*4.80*	4.41	*5.36*	*5.00*

Table 3. Mel-cepstral distances between original and generated speech with random pseudo data set [dB]

Method		FSM	FKS	MHT	MMY	Avg.
Random	Worst	5.14	*4.73*	*4.44*	*5.27*	*4.90*
	Average	5.10	4.61	4.34	5.18	4.81
	Best	**5.09**	**4.54**	**4.28**	**5.11**	**4.76**
K-medoids		*5.16*	4.59	4.42	5.25	4.86

in a reading style taken from ATR Japanese speech database set B [18] were used. All speakers were professional narrators. We chose 450 sentences for training and used remaining 53 sentences as test data. The total amounts of training and test data for respective speakers are shown in Table 1.

The phone boundary information of FSM was annotated manually whereas that of other speakers was given using HMM-based automatic forced alignment. Speech signals were sampled at a rate of 16kHz, and the frame shift was 5ms. In this study, we modeled and generated a spectral features only. 0-39th mel-cepstral coefficients derived from the spectral envelope extracted by STRAIGHT [19] were used as the spectral features. The maximum number of frames B of each block was set to 1000[1]. The maximum number of iterations of the EM algorithm in hyperparameter optimization was 3. The first 50 sentences included in training were used for hyperparameter optimization and pseudo data selection.

For the construction of decision trees, 5-state, left-to-right, no-skip hidden semi-Markov model (HSMM) was used as a model topology. The output distribution in each state was modeled with a single Gaussian pdf, with diagonal covariance matrices. The feature vector included delta and delta-delta dynamic features as well as the static one. The context set including triphone, accent, and sentence length was used for the HMM training. In the decision-tree-based context clustering for parameter tying, MDL was used as a stopping criterion [20].

The predictive distributions of synthetic speech parameters were generated using the phone durations of original utterances and the predictive mean sequences were used for objective evaluation. We also evaluated the HMM-based speech synthesis as a baseline for statistical parametric speech synthesis. The experiments were performed on an Intel Core i7-4770K 3.5GHz CPU and Theano [21] was used for matrix operations.

4.2. Objective evaluation

To evaluate the performance with different pseudo data sets, we calculated mel-cepstral distance, equivalently, root mean square error between the original and generated log spectra on a mel-frequency scale. The results using different pseudo data sizes are shown in Table 2. The results for HMM-based technique are also shown in the table. In the experiment, the k-medoids clustering was used for pseudo data selection. As can be seen from the table, the distortion became the smallest for all speakers when the pseudo data size M is 2048 and tended to decrease with the increase of the pseudo data size. In addition, it is confirmed that the distortions for the GPR-based methods are lower than those of HMM-based one even when the pseudo data size is 256. Table 3 shows the mel-cepstral distances using randomly chosen pseudo data sets. Five different pseudo data set were chosen and their average, worst, and best distortions are shown in the table. It is seen that random choice of pseudo data set, which is simpler than k-medoids-based method, seems to be acceptable because the distances of the worst cases were comparable with those of k-medoids except for the speaker FKS.

4.3. Computational complexity and footprint

As shown in the previous section, the use of more pseudo data points reduces the spectral distortion. However, the increase of pseudo data points causes further computational cost for not only training but also synthesis and larger model footprint. For the realization of a real-time synthesis system, it requires shorter generation time than the utterance duration. To evaluate the trade-off between computational cost and performance, we show computation time and model footprint of the proposed technique.

Tables 4 shows the average computation time for each step of training and synthesis processes. The pseudo data set was divided using the k-medoids clustering. The training was separated into two phases: hyperparameter optimization and GPR parameter training. The processing times at the first iteration in hyperparameter optimization are shown in the table. It is seen that the predominant step in hyperparameter optimization is the M-step in the EM iteration of kernel parameter optimization. This is because the each kernel parameter was picked up and the gradient for the parameter was calculated in each SGD step. The computation time for hyperparameter optimization became 1.3 to 2.4 times when the pseudo data size got twice. Similarly, the processing times for GPR parameter training also increased with the pseudo data size. However, it can be seen that the computation time is shorter than the total length of training data even using 2048 pseudo data points. As for the inference of predictive distribution in the synthesis process, 512 or smaller pseudo points achieved shorter inference time than the actual average utterance length of 3.81 sec.

Table 5 shows the model footprint. The predominant occupation of model footprint is the kernel matrices $\mathbf{K}_{B_s M}$ and $\mathbf{\Lambda}_{B_s}$. The average footprint size can be approximated by a simple linear function $(1.4 \times 10^{-3} M + 1.33)$ GB where M is a pseudo data size.

5. CONCLUSIONS

In this paper, we have described the framework of speech synthesis based on GPR with PIC approximation. Furthermore, we examined the methods of choosing the pseudo data set, and investigated the trade-off between performance and computational cost with multiple sizes of the pseudo data set through experiments under practical conditions. Throughout the experiments using four speakers, it is confirmed that the proposed GPR-based framework outperformed the conventional HMM-based one. Moreover it is found that randomly chosen pseudo data set is acceptable compared with the k-medoids-based method. In future work, we should model F0 contours and

[1]In fact, B is also a hyperparameter of GPR, and it is expected that not only the performance improves but also the computational cost increases if B increases. The effect of the value of B will be examined in future work.

Table 4. Computation time of optimization, training, and synthesis processes. Average processing times for one sentence are shown in the synthesis step [sec].

	Hyperparameter optimization					GPR-parameter training			Synthesis
Pseudo data size	Pseudo data initialization	K-medoids clustering	E-step	M-step	Sum	Kernel matrix computation	GPR-parameter computation	Sum	Predictive distribution
128	67	140	116	1081	1404	461	591	1051	2.54
256	57	131	122	1393	1702	511	612	1123	3.02
512	47	156	137	2318	2657	620	691	1311	3.40
1024	47	222	166	5007	5443	821	661	1482	4.78
2048	52	334	209	11865	12461	1133	1114	2247	7.48

Table 5. Model footprint in GBytes.

Pseudo data size	FSM	FKS	MHT	MMY	Avg.
128	1.52	1.41	1.50	1.33	1.44
256	1.75	1.63	1.72	1.54	1.66
512	2.20	2.05	2.16	1.94	2.09
1024	3.02	2.87	2.99	2.70	2.89
2048	4.23	4.11	4.35	3.92	4.15

phone durations and examine the effectiveness of the proposed GPR-based speech synthesis.

6. REFERENCES

[1] S. Park and S. Choi, "Gaussian process regression for voice activity detection and speech enhancement," in *Proc. IJCNN*, 2008, pp. 2879–2882.

[2] N. C. V. Pilkington, H. Zen, and M. J. F. Gales, "Gaussian process experts for voice conversion," in *Proc. INTERSPEECH*, 2011, pp. 2761–2764.

[3] H. Park and C. D Yoo, "Gaussian process dynamical models for phoneme classification," in *NIPS 2011 Workshop on Bayesian Nonparametrics: Hope or Hype*, 2011.

[4] R. Fernandez, A. Rendel, B. Ramabhadran, and R. Hoory, "F0 contour prediction with a deep belief network-Gaussian process hybrid model," in *Proc. ICASSP*, 2013, pp. 6885–6889.

[5] T. Koriyama, T. Nose, and T. Kobayashi, "Statistical parametric speech synthesis based on Gaussian process regression," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 2, pp. 173–183, 2014.

[6] T. Koriyama, T. Nose, and T. Kobayashi, "Parametric speech synthesis based on Gaussian process regression using global variance and hyperparameter optimization," in *Proc. ICASSP*, 2014, pp. 3862–3866.

[7] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *In NIPS 18, MIT press*, 2006, pp. 1257C–1264.

[8] E. Snelson and Z. Ghahramani, "Local and global sparse Gaussian process approximations," in *Proc. AISTATS*, 2007, pp. 524–531.

[9] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Proc. AISTATS*, 2009, pp. 16–19.

[10] K. P. Murphy, *Machine learning: a probabilistic perspective*, MIT Press, 2012.

[11] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis," in *Proc. EUROSPEECH*, 1999, pp. 2347–2350.

[12] H. Zen, K. Tokuda, and A.W. Black, "Statistical parametric speech synthesis," *Speech Communication*, vol. 51, no. 11, pp. 1039–1064, 2009.

[13] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, MIT press Cambridge, MA, 2006.

[14] P. Dutilleul, "The mle algorithm for the matrix normal distribution," *Journal of statistical computation and simulation*, vol. 64, no. 2, pp. 105–123, 1999.

[15] T. Fukuda and T. Nitta, "Orthogonalized distinctive phonetic feature extraction for noise-robust automatic speech recognition," *IEICE Trans. Inf. & Syst.*, vol. 87, no. 5, pp. 1110–1118, 2004.

[16] J. J. Odell, *The use of context in large vocabulary speech recognition*, Ph.D. thesis, University of Cambridge, 1995.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge Univ. Press, 1992.

[18] A. Kurematsu, K. Takeda, Y. Sagisaka, S. Katagiri, H. Kuwabara, and K. Shikano, "ATR Japanese speech database as a tool of speech recognition and synthesis," *Speech Communication*, vol. 9, no. 4, pp. 357–363, Aug. 1990.

[19] H. Kawahara, I. Masuda-Katsuse, and A. de Cheveigne, "Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds," *Speech Communication*, vol. 27, no. 3-4, pp. 187–207, 1999.

[20] K. Shinoda and T. Watanabe, "MDL-based context-dependent subword modeling for speech recognition," *Acoustical Science and Technology*, vol. 21, no. 2, pp. 79–86, 2000.

[21] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: A CPU and GPU math compiler in Python," in *Proceedings of the Python for Scientific Computing Conference, SciPy*, 2010, pp. 3–10.