Hindawi
Journal of Electrical and Computer Engineering
Volume 2017, Article ID 4890921, 8 pages
https://doi.org/10.1155/2017/4890921

Research Article

Anomaly Detection for Aviation Safety Based on an Improved KPCA Algorithm

Xiaoyu Zhang, Jiusheng Chen, and Quan Gan

College of Electronics, Information & Automation, Civil Aviation University of China, Tianjin 300300, China

Correspondence should be addressed to Xiaoyu Zhang; xy_zhang@cauc.edu.cn

Received 20 August 2016; Accepted 8 March 2017; Published 16 March 2017

Academic Editor: R. Aguilar-López

Thousands of flights datasets should be analyzed per day for a moderate sized fleet; therefore, flight datasets are very large. In this paper, an improved kernel principal component analysis (KPCA) method is proposed to search for signatures of anomalies in flight datasets through the squared prediction error statistics, in which the number of principal components and the confidence for the confidence limit are automatically determined by OpenMP-based K-fold cross-validation algorithm and the parameter in the radial basis function (RBF) is optimized by GPU-based kernel learning method. Performed on Nvidia GeForce GTX 660, the computation of the proposed GPU-based RBF parameter is 112.9 times (average 82.6 times) faster than that of sequential CPU task execution. The OpenMP-based K-fold cross-validation process for training KPCA anomaly detection model becomes 2.4 times (average 1.5 times) faster than that of sequential CPU task execution. Experiments show that the proposed approach can effectively detect the anomalies with the accuracy of 93.57% and false positive alarm rate of 1.11%.

1. Introduction

In a flight, onboard Quick Access Recorder (QAR) and Flight Data Recorder (FDR) can record more than 600 parameters sampled at 1 Hz. Typical FDR and QAR parameters are acquired from the propulsion system, avionics system, landing gear, cockpit switches position, control surfaces, and other critical systems. Due to the complexity of flight conditions and aircraft systems, there are a large number of abnormalities that are often unclear. Detecting precursors of aviation safety incidents based on anomalies detection of FDR and QAR data from aircraft is becoming more and more important for pilots and airline safety management teams. Once these anomalies are detected, the additional contextual information of historical flight data is needed to determine the frequency of anomalies occurrence.

Bay and Schwabacher [1] proposed a distance-based anomaly detection method called Orca. The distance metric was used to detect abnormal points. The output from Orca was a distance score representing the average distance to its k-nearest neighbors. Higher score meant more anomalies since the nearest neighbors were farther away. Iverson et al. [2] developed a health monitoring software called Inductive Monitoring System (IMS). Clustering method was used to analyze system data; a higher composite distance value was adopted to detect anomaly data.

Recently, kernel methods were applied to flight data analysis. Das et al. [3] developed a multiple kernel anomaly detection (MKAD) algorithm which could work with heterogeneous datasets including both continuous and discrete sample sequences. MKAD could detect some significant anomalies from real aviation operational data. The major advantage of MKAD was that it could detect potential safety anomalies in both discrete streams and continuous streams.

Smart et al. [4] presented a two-phase novelty detection approach to locate abnormalities of flight data in the descent phase. Compared with the mixture of Gaussians and K-means, the support vector machine (SVM) provided the best detection rate and identified obstacle abnormalities with higher accuracy.

Kernel principal component analysis (KPCA) algorithm was used by Cho et al. [5] for fault identification in process monitoring. There were two significant problems for anomaly detection in the KPCA-based method: computation performance and adaptability. For the KPCA-based method, the kernel matrix was defined for the principal component

feature extraction and analysis. However, computing the associated kernel matrices required $O(n^3)$ computational complexity, where n was the size of the training data, which limited the applicability of these methods for large dataset. Because the parameter in the kernel function, the number of principal components, and the confidence for the confidence limit should be set before anomaly detection by KPCA method, the adaptability of the KPCA method was largely limited.

To solve problems of KPCA, many extended methods were developed including fast iterative KPCA (FIKPCA) [6], adaptive KPCA [7, 8], and multiscale KPCA [9]. The adaptive KPCA [7] could flexibly track the change of external environment or sample data. However, the model should be updated with real-time data, which affected the real-time diagnosis with increasing data. In [8], the number of principal components and squared prediction error (SPE) confidence limit were obtained by experience.

The method proposed in this paper is based on KPCA and aims at anomaly detection in the flight. In order to analyze vast amounts of flight data, the anomaly detection algorithm should be effectively accelerated. The rest of this paper is organized as follows. In Section 2, classical KPCA algorithm is introduced. Section 3 introduces the improved KPCA algorithm used to optimize key parameters such as the parameter of the radial basis function (RBF), the number of principal components, and the SPE confidence limit. Experimental results and performance evaluation of the proposed anomaly detection model are given in Section 4.

2. Kernel Principal Component Analysis

KPCA is one of kernel-based learning methods, which is the extension of principal component analysis (PCA) in the nonlinear area. Basic KPCA method is introduced as follows [5–11].

Given the training set $D = \{(\mathbf{x}_i, y_i),\ i = 1, \ldots, n\}$, $\mathbf{x}_i \in R^d$, $i = 1, 2, \ldots, n$ is the training data with zero mean, $y_i = \{+1, -1\}$, $i = 1, 2, \ldots, n$ is the class label, and the sample covariance matrix $\mathbf{C}$ in the feature space is expressed as

$$\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n} \phi\left(\mathbf{x}_i\right)\phi\left(\mathbf{x}_i\right)^{\mathrm{T}}, \tag{1}$$

where $\phi(\cdot)$ is a nonlinear mapping function. The principal component can be obtained by

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v} = \frac{1}{n}\sum_{i=1}^{n} \left\langle \phi\left(\mathbf{x}_i\right), \mathbf{v}\right\rangle \phi\left(\mathbf{x}_i\right), \tag{2}$$

where $\langle \mathbf{x}, \mathbf{y}\rangle$ denotes the dot product of $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{v} \neq \mathbf{0}$ is the eigenvector, and $\lambda > 0$ is the corresponding eigenvalue. The $\mathbf{v}$ corresponding to the largest λ is the first principal component, and the $\mathbf{v}$ corresponding to the smallest λ is the last principal component. The eigenvector $\mathbf{v}$ can be expressed as

$$\mathbf{v} = \sum_{i=1}^{n} \alpha_i \phi\left(\mathbf{x}_i\right). \tag{3}$$

To obtain coefficients α_i, $i = 1, \ldots, n$, defining the kernel matrix $\mathbf{K}$ of $n \times n$, the elements in the matrix are determined by $K_{ij} = \phi(\mathbf{x}_i)^{\mathrm{T}}\phi(\mathbf{x}_j) = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, where $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ is the inner product of $\mathbf{x}_i$ and $\mathbf{x}_j$. In this paper, the kernel matrix $\mathbf{K}$ is calculated through RBF kernel function, which is adopted as follows:

$$\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\beta\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right), \quad 0 < \beta < +\infty, \tag{4}$$

where β is the parameter of RBF. The coefficient $\boldsymbol{\alpha}$ can be obtained by

$$n\lambda\boldsymbol{\alpha} = \mathbf{K}\boldsymbol{\alpha}, \quad \boldsymbol{\alpha} = \left[\alpha_1, \ldots, \alpha_n\right]^{\mathrm{T}}. \tag{5}$$

The centered kernel matrix $\overline{\mathbf{K}}$ can be calculated by

$$\overline{\mathbf{K}} = \mathbf{K} - \mathbf{E}\mathbf{K} - \mathbf{K}\mathbf{E} + \mathbf{E}\mathbf{K}\mathbf{E}, \tag{6}$$

where matrix $\mathbf{E} = (1/n)\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix} \in R^{n\times n}$.

Assuming $\phi(\mathbf{x}_{\mathrm{new}})$ is a nonlinear mapping function that projects the new data $\mathbf{x}_{\mathrm{new}} \in R^d$ from the original space to the feature space, the projection from $\phi(\mathbf{x}_{\mathrm{new}})$ onto the eigenvector $\mathbf{v}_l$, $l = 1, 2, \ldots, p$; $1 \leq p \leq n$ can be obtained by

$$\begin{aligned} \mathbf{t}_l &= \left\langle \mathbf{v}_l, \phi\left(\mathbf{x}_{\mathrm{new}}\right)\right\rangle = \sum_{i=1}^{n} \boldsymbol{\alpha}_i^l \left\langle \phi\left(\mathbf{x}_i\right), \phi\left(\mathbf{x}_{\mathrm{new}}\right)\right\rangle \\ &= \sum_{i=1}^{n} \boldsymbol{\alpha}_i^l \kappa\left(\mathbf{x}_i, \mathbf{x}_{\mathrm{new}}\right). \end{aligned} \tag{7}$$

The number of principal components can influence the detection rate and computational complexity. In the anomaly detection model, the cumulative percent variance theory [8] is adopted to calculate the number of principal components:

$$\mathrm{CPV}\left(p\right) = \frac{\sum_{i=1}^{p}\lambda_i}{\sum_{i=1}^{n}\lambda_i} \geq \gamma, \tag{8}$$

where γ is the threshold of principal components and p is the number of principal components.

SPE statistic is adopted to detect abnormal flight data in this paper. In [11], SPE is expressed as follows:

$$\begin{aligned} \mathrm{SPE} &= \left\|\phi\left(\mathbf{x}\right) - \widehat{\phi}_p\left(\mathbf{x}\right)\right\|^2 = \left\|\widehat{\phi}_n\left(\mathbf{x}\right) - \widehat{\phi}_p\left(\mathbf{x}\right)\right\|^2 \\ &= \sum_{i=1}^{n} t_i^2 - \sum_{i=1}^{p} t_i^2. \end{aligned} \tag{9}$$

The confidence limit for SPE is calculated through its approximate distribution [11]:

$$\mathrm{SPE}_\eta \sim g\chi_h^2, \tag{10}$$

where η is the confidence degree of χ^2 distribution, $g = b/2a$, $h = 2a^2/b$, a is the estimated mean of the SPE, and b is the estimated variance of the SPE.

3. Improved KPCA Anomaly Detection Model

To improve the computation efficiency of KPCA-based method, an improved KPCA method based on parallel computing is proposed, where the RBF parameter is calculated

Input. $D = \{(\mathbf{x}_i, y_i),\ i = 1, \ldots, n\} \in R^d \times Y,\ Y = \{+1, -1\}$: the training set
Output. $\beta^* \in R$: the optimal RBF parameter
(1) $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
(2) $\mathbf{\Omega}_{ij} = -\|\mathbf{x}_i - \mathbf{x}_j\|^2,\ i, j = 1, \ldots, n$
(3) $\widetilde{\mathbf{K}}_{ij} = \begin{cases} 1, & \text{if } y_i = y_j \\ 0, & \text{if } y_i \neq y_j \end{cases}$
(4) **repeat**
(5) $J_\beta = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\widetilde{\mathbf{K}}_{ij} - \mathbf{K}_{ij})^2$
(6) $\nabla_\beta J = \sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{x}_i - \mathbf{x}_j\|^2(\widetilde{\mathbf{K}}_{ij} - \mathbf{K}_{ij})\mathbf{K}_{ij}$
(7) $\beta = \beta - \delta\nabla_\beta J$
(8) **until** convergence

Algorithm 1: Gradient descent algorithm for obtaining the optimal RBF parameter.

on GPU and implemented by CUDA [12], and the number of principal components and the confidence for the confidence limit are determined via parallel K-fold cross-validation by OpenMP [13].

3.1. RBF Parameter Computation by CUDA. Because the structure of the training data in the feature space is determined by the chosen kernel function, inappropriate choice of parameter of kernel function will seriously affect the detection performance. In this paper, an optimized method for calculating the parameter of the RBF kernel function is proposed based on the training data.

Two important properties of the RBF kernel are $\kappa(\mathbf{x}, \mathbf{x}, \beta) = 1$ and $0 < \kappa(\mathbf{x}, \mathbf{z}, \beta) \leq 1$. The optimization of the RBF parameter β can be achieved by following properties [14]:

(1) For samples in the same class, the RBF kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j, \beta) \approx 1$ if $y_i = y_j$.
(2) For samples in different classes, the RBF kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j, \beta) \approx 0$ if $y_i \neq y_j$.

The optimal RBF parameter β^* is obtained by solving the following optimization problem [15]:

$$\min_{\beta>0} \quad J(\beta) \equiv \left\|\widetilde{\mathbf{K}} - \mathbf{K}\right\|_2^2 = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\widetilde{\mathbf{K}}_{ij} - \mathbf{K}_{ij}\right)^2, \tag{11}$$

where $\widetilde{\mathbf{K}}$ is a constant matrix $\widetilde{\mathbf{K}} = (\widetilde{\mathbf{K}}_{ij})_{i,j=1}^n$ with entries

$$\widetilde{\mathbf{K}}_{ij} = \begin{cases} 1, & \text{if } y_i = y_j \\ 0, & \text{if } y_i \neq y_j. \end{cases} \tag{12}$$

$\mathbf{K}$ is the kernel matrix $\mathbf{K} = (\mathbf{K}_{ij})_{i,j=1}^n$ with entries

$$\mathbf{K}_{ij} = \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\beta\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right), \quad i, j = 1, \ldots, n. \tag{13}$$

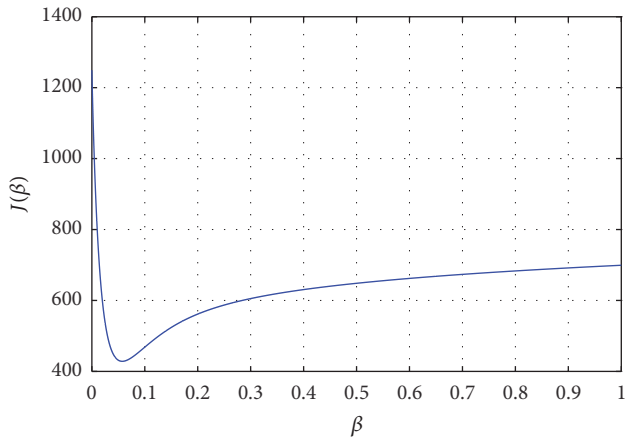

Figure 1: Schematic illustration of the function $J(\beta)$.

The schematic illustration of function $J(\beta)$ is shown in Figure 1. The horizontal axis represents the RBF parameter β and the vertical axis represents the corresponding $J(\beta)$. The graph shows that there is only one minimum value which is the optimal RBF parameter value in the proposed method.

The optimal RBF parameter β^* is obtained via gradient descent [16]. The partial derivative of function $J(\beta)$ is expressed as

$$\frac{\partial}{\partial\beta}J(\beta) = \sum_{i=1}^{n}\sum_{j=1}^{n}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\left(\widetilde{\mathbf{K}}_{ij} - \mathbf{K}_{ij}\right)\mathbf{K}_{ij}. \tag{14}$$

The iterative formula of gradient descent is expressed as follows:

$$\beta = \beta - \delta\nabla_\beta J, \tag{15}$$

where $\nabla_\beta J$ is the partial derivative of function $J(\beta)$ and δ is the learning step. The gradient descent algorithm for obtaining the optimal RBF parameter is given in Algorithm 1.

Input. $D = \{(\mathbf{x}_i, y_i),\ i = 1, \ldots, n\} \in R^d \times Y,\ Y = \{+1, -1\}$: the training set
Output. $\beta^* \in R$: the optimal RBF parameter
(1) $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
(2) $\mathbf{\Omega}_{ij} = -\|\mathbf{x}_i - \mathbf{x}_j\|^2,\ i, j = 1, \ldots, n$
(3) $\widetilde{\mathbf{K}}_{ij} = \begin{cases} 1, & \text{if } y_i = y_j \\ 0, & \text{if } y_i \neq y_j \end{cases}$
(4) Transfer $\mathbf{\Omega}$ from CPU to GPU
(5) Transfer $\widetilde{\mathbf{K}}$ from CPU to GPU
(6) **repeat**
(7) (GPU) $J_\beta = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=i}^{n}(\widetilde{\mathbf{K}}_{ij} - \mathbf{K}_{ij})^2$
(8) (GPU) $\nabla_\beta J = \sum_{i=1}^{n}\sum_{j=i}^{n}\|\mathbf{x}_i - \mathbf{x}_j\|^2(\widetilde{\mathbf{K}}_{ij} - \mathbf{K}_{ij})^2\mathbf{K}_{ij}$
(9) $\beta = \beta - \delta\nabla_\beta J$
(10) **until** convergence

Algorithm 2: GPU code of gradient descent algorithm for obtaining the optimal RBF parameter.

The approach is executed with the sequential algorithm on CPU. When the amount of sample increases, computation time increases rapidly. If the dimension of the kernel matrix is 8350, the total computation time increases to 1106 s. The computation of lines (5) and (6) in Algorithm 1 is implemented through level 2 for-loop, so it is better to perform them on GPU. The optimal RBF parameter β^* is obtained by a parallel algorithm on GPU which replaces overlapped ones by the sequential CPU algorithm. The GPU code of the gradient descent algorithm for obtaining the optimal RBF parameter is given in Algorithm 2.

If kernel matrix $\mathbf{K}$ is transferred from GPU to CPU in each step of the convergence loop, the data transfer becomes an acceleration bottleneck. To reduce the amount of transferred data, we focus on the structure of matrices $\mathbf{\Omega}$ and $\widetilde{\mathbf{K}}$ as described in lines (2) and (3) of Algorithm 2.

The matrix $\mathbf{\Omega}$ and matrix $\widetilde{\mathbf{K}}$ do not change with the optimal RBF parameter β^* in each step of the convergence loop; the transfer of matrices $\mathbf{\Omega}$ and $\widetilde{\mathbf{K}}$ is necessary only in the initialization. In addition, $\mathbf{\Omega}$ and $\widetilde{\mathbf{K}}$ are symmetric matrices. Thus, it is necessary to transfer upper triangular part of matrices $\mathbf{\Omega}$ and $\widetilde{\mathbf{K}}$ from CPU to GPU. The size of the transferred data in Algorithm 2 on lines (4) and (5) is $((n + 1) \times n)/2$, respectively.

3.2. Parameters γ and η Computation by OpenMP. To make the anomaly detection model more accurate, the number of principal components and the confidence for the confidence limit are determined via K-fold cross-validation [20].

The K-fold cross-validation process randomly splits $\mathbf{D}$ into K disjointed parts with almost equal size $D_1, \ldots, D_K$. At each kth fold, $\mathbf{D}_k$ $(k = 1, \ldots, K)$ and $\mathbf{D}^{(-k)} = \mathbf{D} - \mathbf{D}_k$ are used as the test dataset and the training dataset. To assess the performance of the K-fold cross-validation, the balanced error rate matrix is utilized. The balanced error rate (BER) [4] is given by

$$\text{BER} = \text{FPR} + \text{FNR}, \tag{16}$$

where false positive rate (FPR) denotes the percentage of incorrectly identified normal flight data and false negative rate (FNR) denotes the percentage of incorrectly identified abnormal flight data

The threshold of principal components γ and the confidence for the confidence limit η are confirmed according to a fivefold cross-validation with $\log_2^{1-\gamma} \in \{-7, -6, \ldots, -1\}$ and $\log_2^{1-\eta} \in \{-7, -6, \ldots, -1\}$. The K-fold cross-validation process can be exploited by loop-level parallelism which executes loop iterations across multiple processors concurrently. The execution time can be reduced significantly. The OpenMP-based K-fold cross-validation algorithm for obtaining optimal parameters γ^* and η^* is given in Algorithm 3.

4. Numerical Experiments and Discussions

The performance evaluation of the proposed algorithm in the CPU-GPU heterogeneous environment is given in this section. For the GPU algorithm, i7-3770 CPU with Nvidia GTX 660 is used, and the bandwidth is about 144.2 GB/s. For the CPU algorithm i7-3770@3.40 GHz CPU with 8 GB SDRAM is used. Synthetic datasets [3] are used in the experiment, which include 300 flights, each flight with 1000 sample points. In this paper, fault type I in the synthetic data is used to access the performance of the improved KPCA. The specifications of the experiment environment are listed in Table 1.

4.1. Computation Performance. Experiment results are showed in Figures 2, 3, and 4. Figure 2 shows the dimension of the kernel matrix, elapsed time for computing the parameter of RBF function on GPU and CPU, and speedup achieved by the parallel code over the sequential code. Figure 3 shows the breakdown of elapsed time for computing the parameter of RBF function on GPU. Figure 4 shows the dimension of the kernel matrix, elapsed time for computing the K-fold cross-validation process, and speedup achieved by the parallel code over the sequential code.

Input. $\mathbf{D} = \{(\mathbf{x}_i, y_i),\ i = 1, \ldots, n\} \in R^d \times Y,\ Y = \{+1, -1\}$: the training set, K: const of K-fold
Output. γ^*, η^*: optimal parameters
(1) Partiton($\mathbf{D}, K$)
(2) #pragma omp parallel for schedule (dynamic)
(3) **for** $i = 1$ to K **do**
(4) $\quad \mathbf{T} = \mathbf{D} - \mathbf{D}_i$
(5) $\quad$ **for** $m = 1$ to 7 **do**
(6) $\quad\quad$ **for** $n = 1$ to 7 **do**
(7) $\quad\quad$ Training($\mathbf{T}, \gamma_m, \eta_n$)
(8) $\quad\quad$ Testing($\mathbf{D}_i, \gamma_m, \eta_n$)
(9) $\quad\quad$ $\mathrm{BER}_{m,n} = \mathrm{FPR}_{m,n} + \mathrm{FNR}_{m,n}$
(10) $\quad\quad$ **end for**
(11) $\quad$ **end for**
(12) **end for**

Algorithm 3: OpenMP-based K-fold cross-validation algorithm for obtaining optimal parameters γ^* and η^*.

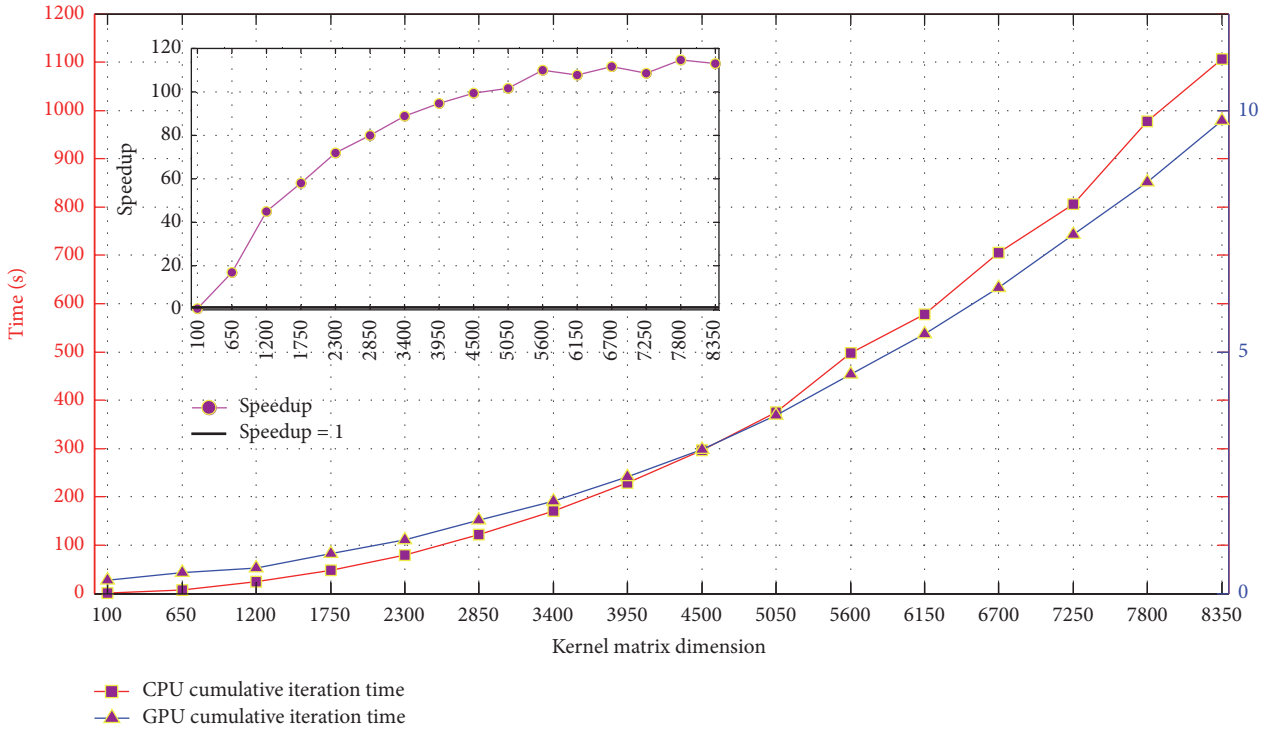

Figure 2: Elapsed time and speedup for RBF parameter computation on GPU and CPU.

Table 1: Specifications of the experiment environment.

CPU	Intel(R) Core(TM) i7-3770 (3.4 GHz, 4 core) Memory: DDR3 SDRAM 8 GB
GPU	NVIDIA GeForce GTX 660 Memory: GDDR5 2 GB
Software	CUDA Toolkit 5.0 [12] OpenMP 2.0 [13] Armadillo 4.0 [17] mlpack-1.0 [18], using the data load function and gradient descent function gsl 1.8 [19], using the gsl_cdf_chisq_Pinv function

As shown in Figure 2, the speedup on GPU code is larger than 100 when the dimension of the kernel matrix is larger than 5050. Elapsed time on GPU is slower than that on CPU with small dimension of kernel matrix, but faster with large kernel matrix dimension. When the dimension of kernel matrix is 100, the elapsed time on GPU is up to 2.3 times slower than that on CPU. However, it is up to 112.9 times faster on GPU than that on CPU while the dimension of kernel matrix is 8350. The reason is that when kernel matrix dimension is small, the synchronization cost for parallelism on GPU is larger than that needed on CPU and the data transfer between the CPU and GPU becomes the accelerating

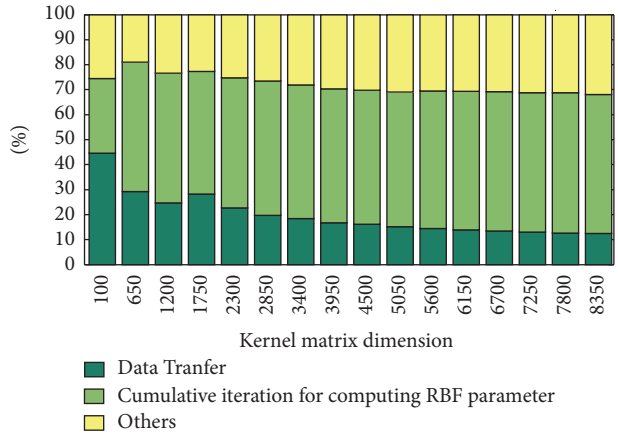

Figure 3: Breakdown of elapsed time for RBF parameter computation on GPU.

bottleneck. When the dimension of the kernel matrix is small, the computation cost of the exponentiation and addition arithmetic is negligible on CPU, and the effective acceleration by parallel processing cannot offset the data transfer time between the CPU and GPU.

As shown in Figure 3, the ratio of time taken up by "Data Transfer" and "Others" is relatively high for small-dimension kernel matrix. As the dimension of the kernel matrix becomes larger, computation cost becomes higher, and almost all the elapsed time is spent on exponentiation and addition arithmetic for RBF parameter computation, so the time for data transfer is relatively reduced.

The time for "Data Transfer" is significantly lower than that in other processes when the dimension of the kernel matrix is larger. For large-dimension kernel matrices, the occupancy of "Data Transfer" is less than 15%. In other words, data transfer does not become the acceleration bottleneck and the size of data transferred between the CPU and GPU is effectively reduced in the proposed GPU-based method, especially for large-dimension matrices.

As shown in Figure 4, the running time of the parallel algorithm and the sequential algorithm increases quickly when the kernel matrix dimension increases. When the dimension of the kernel matrix is 100, the parallel algorithm is 1.8 times faster than the sequential algorithm; when the dimension of the kernel matrix is 500, it is up to 2.4 times faster, but 1.1 times slower when the dimension of the kernel matrix is 1500. The reason is that the OpenMP-based parallel algorithm needs to synchronize among multiple processors. In this case, when kernel matrix dimension is large, the synchronization cost on parallel algorithm becomes higher than that needed on sequential algorithm.

4.2. Detection Performance. In the synthetic datasets [3], a set of Gaussian distributions was defined for each continuous parameter. The continuous parameters would draw from their defined distribution at each time step. The statistical distribution of each continuous parameter is shown in Figure 5. It can be clearly seen that the statistical distribution of each continuous parameter follows Gaussian distribution, and favorable results are obtained by normal probability density function.

The initial training data number (including normal sample and abnormal sample), the test sample number, the best parameters, the detection rate (DR), FNR, and FPR are listed in Table 2. The continuous parameters are decomposed for three layers through Daubechies-4 (DB4) wavelet. Wavelet coefficients of the third layer are used to reconstruct high-frequency bands and scaling coefficients of the third layer are used to reconstruct low-frequency bands. Then, the feature parameter vector is constructed by the high-frequency and low-frequency bands of the continuous parameters. DR denotes the percentage of correctly identified abnormal flight data, FPR denotes the percentage of incorrectly identified normal flight data, and FNR denotes the percentage of incorrectly identified abnormal flight data. From the result, when normal sample is 1000 and abnormal sample is 266, DR is up to 93.57% with FPR of 1.11%.

4.3. Comparison of Some Detection Methods. Two kinds of kernel-based fault detection methods, one-class support vector machine (OCSVM) [21] and improved KPCA, are compared, and results are listed in Table 3. For both algorithms, the number of initial samples and testing samples are 1266 (1000 normal samples and 266 abnormal samples) and 10000, respectively.

For OCSVM, parameter ν of OCSVM is searched in $\{0.01k, 0.1k\}$, where $k = 1, 2, \ldots, 9$, and parameter β of the RBF kernel is searched in 2^{2k-1}, where $k = -7, -6, \ldots, 3$. The parameters ν and β are selected by fivefold cross-validation. LIBSVM is used for OCSVM detection model training [22].

For improved KPCA, parameter β of the RBF kernel is adjusted by the gradient descent algorithm, which is given in Algorithm 1. The parameters γ and η are searched according to a fivefold cross-validation with $\log_2^{1-\gamma} \in \{-7, -6, \ldots, -1\}$ and $\log_2^{1-\eta} \in \{-7, -6, \ldots, -1\}$, respectively.

From Table 3, it can be clearly seen that the FPR of the improved KPCA fault detection method is better than that of OCSVM. But the DR and FNR of OCSVM fault detection method are better than those of the improved KPCA. Considering DR, FNR, and FPR, the detection performance of the improved KPCA fault detection method is more favorable compared to that of OCSVM.

5. Conclusion

Kernel method is often used in anomaly detection in the field of aviation safety. The computation performance is the main problem in aviation data analysis domain. In this paper, an improved KPCA solution is proposed for efficient anomaly detection. The RBF parameter is optimized by GPU and OpenMP-based K-fold cross-validation is adopted for training KPCA anomaly detection model. The experiment was performed with i7-3770 CPU and Nvidia GTX 660. Results show that RBF parameter computation accelerates up

Table 2: Detection performance by the improved KPCA method.

Training data		Test sample number	Best Parameters	DR (%)	FNR (%)	FPR (%)
Normal sample number	Abnormal sample number					
600	266	10000	$\beta = 0.031, \alpha = 0.75, \eta = 0.9921875$	93.52	6.48	1.16
800	266	10000	$\beta = 0.0245, \alpha = 0.5, \eta = 0.9921875$	93.77	6.23	0.91
1000	266	10000	$\beta = 0.0201, \alpha = 0.5, \eta = 0.9921875$	93.57	6.43	1.11
1200	266	10000	$\beta = 0.0173, \alpha = 0.5, \eta = 0.9921875$	93.41	6.59	1.27
1400	266	10000	$\beta = 0.031, \alpha = 0.75, \eta = 0.9921875$	93.47	6.53	1.21
1600	266	10000	$\beta = 0.0508, \alpha = 0.5, \eta = 0.9921875$	94.34	5.66	0.34

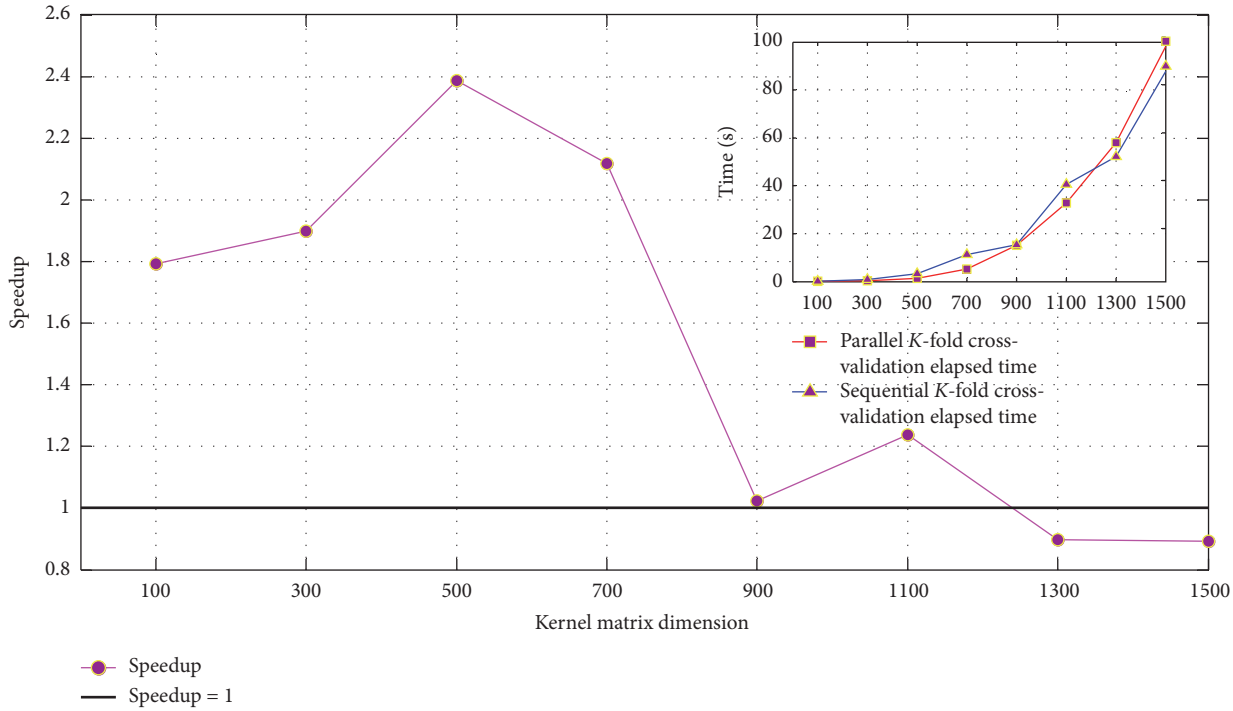

Figure 4: Elapsed time and speedup for K-fold cross-validation.

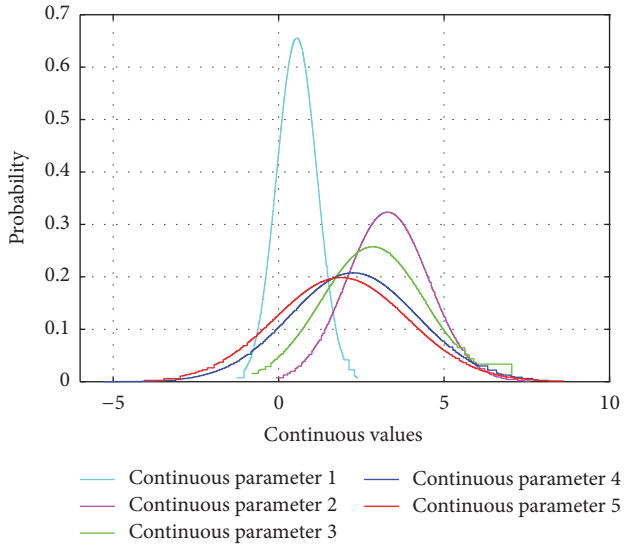

Figure 5: Statistical distribution of continuous parameters.

Table 3: Comparison of different methods.

Algorithm	DR (%)	FNR (%)	FPR (%)
OCSVM	94.55	5.45	21.57
Improved KPCA	93.57	6.43	1.11

to 112.9 times (average 82.6 times) faster by GPU than that by sequential CPU, and training KPCA anomaly detection model accelerates up to 2.4 times (average 1.5 times) faster by OpenMP K-fold cross-validation than that by sequential CPU execution.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was jointly funded by the National Natural Science Foundation of China (Grant no. 61603395) and the National

Natural Science Foundation of China and the Civil Aviation Administration of China (Grants nos. U1433103 and U1533017). The authors would like to express their gratitude for the support provided.

References

[1] S. D. Bay and M. Schwabacher, "Mining distance-based outliers in near linear time with randomization and a simple pruning rule," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, pp. 29–38, ACM, August 2003.

[2] D. L. Iverson, R. Martin, M. Schwabacher et al., "General purpose data-driven system monitoring for space operations," *Journal of Aerospace Computing, Information and Communication*, vol. 9, no. 2, pp. 26–44, 2012.

[3] S. Das, B. L. Matthews, A. N. Srivastava, and N. C. Oza, "Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*, pp. 47–55, July 2010.

[4] E. Smart, D. Brown, and J. Denman, "A two-phase method of detecting abnormalities in aircraft flight data and ranking their impact on individual flights," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1253–1265, 2012.

[5] J.-H. Cho, J.-M. Lee, S. W. Choi, D. Lee, and I.-B. Lee, "Fault identification for process monitoring using kernel principal component analysis," *Chemical Engineering Science*, vol. 60, no. 1, pp. 279–288, 2005.

[6] W. Liao, A. Pizurica, W. Philips, and Y. Pi, "A fast iterative kernel PCA feature extraction for hyperspectral images," in *Proceedings of the 17th IEEE International Conference on Image Processing (ICIP '10)*, pp. 1317–1320, IEEE, Hong Kong, September 2010.

[7] M. Ding, Z. Tian, and H. Xu, "Adaptive kernel principal analysis for online feature extraction," *World Academy of Science, Engineering and Technology*, vol. 59, pp. 288–293, 2009.

[8] J. Ni, C. Zhang, and S. X. Yang, "An adaptive approach based on KPCA and SVM for real-time fault diagnosis of HVCBs," *IEEE Transactions on Power Delivery*, vol. 26, no. 3, pp. 1960–1971, 2011.

[9] Y. Zhang and C. Ma, "Fault diagnosis of nonlinear processes using multiscale KPCA and multiscale KPLS," *Chemical Engineering Science*, vol. 66, no. 1, pp. 64–72, 2011.

[10] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[11] J.-M. Lee, C. Yoo, S. W. Choi, P. A. Vanrolleghem, and I.-B. Lee, "Nonlinear process monitoring using kernel principal component analysis," *Chemical Engineering Science*, vol. 59, no. 1, pp. 223–234, 2004.

[12] NVIDIA, *CUDA Toolkit 5.0 Library*, 2012, https://developer.nvidia.com/cuda-toolkit-50-archive.

[13] OpenMP Architecture Review Board, *OpenMP Application Program Interface, V. 2.0*, OpenMP Architecture Review Board, 2002.

[14] C.-H. Li, C.-T. Lin, B.-C. Kuo, and H.-S. Chu, "An automatic method for selecting the parameter of the RBF kernel function to support vector machines," in *Proceedings of the 30th IEEE International Geoscience and Remote Sensing Symposium (IGARSS '10)*, pp. 836–839, IEEE, Honolulu, Hawaii, USA, July 2010.

[15] X. Zhang, Z. Wu, J. Chen, and M. Yue, "An adaptive KPCA approach for detecting LDoS attack," *International Journal of Communication Systems*, vol. 30, no. 4, pp. 1–11, 2017.

[16] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186, Physica, 2010.

[17] C. Sanderson, "Armadillo: an open source C++ linear algebra library for fast prototyping and computationally intensive experiments," Technical Report Version, NICTA, Sydney, Australia, 2010.

[18] R. R. Curtin, J. R. Cline, N. P. Slagle et al., "MLPACK: a scalable C++ machine learning library," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 801–805, 2013.

[19] M. Galassi, J. Davies, J. Theiler et al., *GNU Scientific Library Reference Manual*, 2015, http://www.gnu.org/software/gsl.

[20] C. Hu, B. D. Youn, and P. Wang, "Ensemble of data-driven prognostic algorithms with weight optimization and k-fold cross validation," in *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE '10)*, pp. 1023–1032, American Society of Mechanical Engineers, August 2010.

[21] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems—NIPS 1999*, pp. 582–588, MIT Press, 1999.

[22] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, article 27, 2011.