*sensors*

MDPI

*Article*

# A Weighted Spatial-Spectral Kernel RX Algorithm and Efficient Implementation on GPUs

**Chunhui Zhao *, Jiawei Li, Meiling Meng and Xifeng Yao**

College of Information and Communication Engineering, Harbin Engineering University, Harbin 150001, China; hgcljw@gmail.com (J.L.); meilingmeng@hrbeu.edu.cn (M.M.); xf.yao1020@gmail.com (X.Y.)

* Correspondence: zhaochunhui@hrbeu.edu.cn; Tel.: +86-451-8258-9810

Academic Editor: Ram M. Narayanan

Received: 22 January 2017; Accepted: 20 February 2017; Published: 23 February 2017

**Abstract:** The kernel RX (KRX) detector proposed by Kwon and Nasrabadi exploits a kernel function to obtain a better detection performance. However, it still has two limits that can be improved. On the one hand, reasonable integration of spatial-spectral information can be used to further improve its detection accuracy. On the other hand, parallel computing can be used to reduce the processing time in available KRX detectors. Accordingly, this paper presents a novel weighted spatial-spectral kernel RX (WSSKRX) detector and its parallel implementation on graphics processing units (GPUs). The WSSKRX utilizes the spatial neighborhood resources to reconstruct the testing pixels by introducing a spectral factor and a spatial window, thereby effectively reducing the interference of background noise. Then, the kernel function is redesigned as a mapping trick in a KRX detector to implement the anomaly detection. In addition, a powerful architecture based on the GPU technique is designed to accelerate WSSKRX. To substantiate the performance of the proposed algorithm, both synthetic and real data are conducted for experiments.

**Keywords:** anomaly detection; graphics processing units (GPUs); hyperspectral imaging; kernel mapping; spatial-spectral information; parallel processing

## 1. Introduction

Hyperspectral imagery (HSI) is served as a three-dimensional cube which contains of two spatial dimensions and one spectral dimension. It provides the ability to distinguish the differences of ground-object spectra, so it has a wide range of applications in target detection [1]. Based on the availability of the prior information, the target detection algorithms can be divided into unsupervised and supervised ones. As accurate prior knowledge is difficult to obtain, unsupervised algorithms (anomaly detection algorithms) have drawn wide interest. Anomaly detection uses the differences between targets and the backgrounds to detect anomalies [2–7].

A widely used anomaly detection algorithm is the RX detector proposed by Reed and Yu [8]. The RX detector is a constant false alarm rate anomaly detection operator based on the assumption that hyperspectral image conforms to the multivariate Gaussian distribution. Actually, it determines a Mahalanobis distance between the pixel under test (PUT) and the background. Since the RX detector only makes use of the low-order statistics of hyperspectral data, it generally outputs an undesirable detection result when the distribution of ground materials is complex. To address this issue, the kernel RX (KRX) algorithm, a nonlinear version of the RX detector, was proposed by Kwon and Nasrabadi [9]. The KRX algorithm can utilize the nonlinear statistical information among hyperspectral bands effectively by mapping the spectral signal of original space into the high-dimensional feature space. In this way, it possesses more desirable detection accuracy compared to the RX algorithm. Unfortunately, the KRX detector is computationally very expensive due to the abundant non-linear kernel functions and inverse covariance matrices. This will affect the detection performance when the background kernel matrix degrades if background data are contaminated by anomalous pixels. Fortunately, reasonable use of spatial information and parallelism can solve the above problems, respectively.

With the increasing spatial resolution, spatial information has played a positive role in the field of hyperspectral images processing. It is critical to use spatial and spectral information for the effective increase of the detection performance in hyperspectral data [10]. Therefore, spatial information has been used for many algorithms, such as LSAD [11], DMSR [12], MD-L [13]. Generally, the method of combined spatial and spectral information has high computational complexity.

Many fast computational methods are used to accelerate hyperspectral image processing because of the high dimensionality of HSI [14]. Recently, with the continuous development of the Graphics Processing Units (GPUs), the powerful general-purpose computing power is revealed, which brings a new idea for the parallel optimization of anomaly detection. The attention of parallel processing for HSI on GPUs is increased [15–18]. The applications on GPUs is studied in accelerating target detection algorithm of Orthogonal Subspace Projection (OSP), and the hyperspectral remote sensing data is divided into blocks, which obtains a better performance [15]. GPUs implementation of hyperspectral anomaly detection based on a multivariate normal mixture model is proposed [17] and the experiment proves that the algorithm can be improved in terms of computational effectiveness.

In this paper, we develop a weighted spatial-spectral kernel RX (WSSKRX) algorithm and its efficient parallel version on GPUs. The proposed algorithm reconstructs the central pixel using the spatial neighborhood information and effectively reduces the interference of the abnormal pixel mixed in the background information. Then, WSSKRX achieves the spatial-spectral information integration by combining the original pixel with reconstructed pixel effectively. In addition, the parallel architecture is designed according to the characteristics of WSSKRX. This architecture is efficiently implemented on GPUs via the Compute Unified Device Architecture (CUDA) from NVIDIA. Experimental results show that the WSSKRX algorithm has better detection accuracy, and a much higher speedup is achieved compared to the Central Processing Unit (CPU) version of WSSKRX detector.

## 2. Methods

### 2.1. Anomaly Detection Methods

RX detector is the classical anomaly detection algorithm in hyperspectral imaging. Generally, there are two typical RX detector variants [19]: the global RX (GRX) detector and local RX (LRX) detector. The KRX detector obtains a better detection performance by exploiting the abundant nonlinear information among hyperspectral bands. In what follows, we review the RX and KRX algorithms.

#### 2.1.1. RX Detector

The RX detector, developed by Reed and Yu, is widely used in anomaly detection. Assume that $\mathbf{x}_i$ is the data sample vector and denoted by $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iL})^{\mathrm{T}}$ where $L$ is the total number of spectral bands. The two hypotheses model is given by:

$$\begin{cases} \mathbf{H}_0: \mathbf{x}=\mathbf{b} & \text{(Target absent)} \\ \mathbf{H}_1: \mathbf{x}=a\mathbf{s}+\mathbf{b} & \text{(Target present)} \end{cases} \tag{1}$$

where $a = 0$ under $\mathbf{H}_0$ and $a > 0$ under $\mathbf{H}_1$. $\mathbf{b}$ is the background clutter, and $\mathbf{s}$ denotes the spectral signature of the target. The model assumes that the data arise from two normal probability density functions with the same covariance matrix but different means. In the $\mathbf{H}_0$ case, the background data are modeled as $N(\boldsymbol{\mu},\mathbf{K})$, and in the $\mathbf{H}_1$ case, the data are modeled as $N(\boldsymbol{\mu} + \mathbf{s}, \mathbf{K})$. The RX detector is defined by the following expression:

$$\delta^{\mathrm{RXD}}(\mathbf{x}_n) = (\mathbf{x}_n - \boldsymbol{\mu})^{\mathrm{T}}\mathbf{K}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}) \tag{2}$$

where $\mu$ is the mean of the background clutter data and $\mathbf{K}$ is the background covariance matrix. $\mathbf{K}$ and $\mu$ are defined by:

$$\mathbf{K} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}(i) - \mu)(\mathbf{x}(i) - \mu)^{\mathrm{T}} \tag{3}$$

$$\mu = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}(i) \tag{4}$$

### 2.1.2. Kernel RX Detector

The kernel RX (KRX) detector is a nonlinear version of the RX anomaly detection. As shown in Figure 1, the algorithm non-linearly maps the inputting signal to the high-dimensional feature space via kernel function, so the linear inseparable parts in original space can be separated linearly. Therefore, KRX algorithm has a better separation performance between background and target.

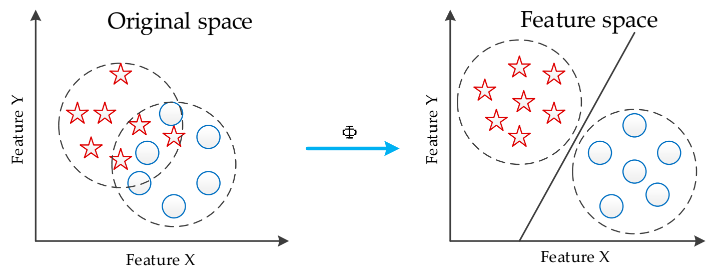


**Figure 1.** Illustration of nonlinear mapping.

The original hyperspectral data $\mathbf{X}_b = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_M]$ is mapped to the high-dimensional feature space by a non-linear function $\Phi$, $\Phi\mathbf{X}_b = [\Phi(\mathrm{x}_1), \Phi(\mathrm{x}_2), \ldots, \Phi(\mathrm{x}_M)]$ is obtained. Then the corresponding KRX algorithm in the feature space is specified by:

$$\mathrm{KRX}(\Phi(\mathbf{r})) = (\Phi(\mathbf{r}) - \hat{\mu}_{b\Phi})^{\mathrm{T}}\hat{\mathbf{K}}_{b\Phi}^{-1}(\Phi(\mathbf{r}) - \hat{\mu}_{b\Phi}) \tag{5}$$

where $\hat{\mathbf{K}}_{b\Phi}$ and $\hat{\mu}_{b\Phi}$ are the covariance matrix and mean of the background samples in the feature space, respectively. The centered inputting matrix in the feature space is expressed as $\mathbf{X}_{\Phi c} = [\Phi_c(\mathrm{x}_1)\Phi_c(\mathrm{x}_2), \ldots, \Phi_c(\mathrm{x}_M)]$, where $\Phi_c(\mathrm{x}_i) = \Phi(\mathrm{x}_i)\ \hat{\mu}_{b\Phi}$. Therefore, the covariance matrix can be represented by $\hat{\mathbf{K}}_{b\Phi} = \frac{1}{M}\mathbf{X}_{\Phi c}\mathbf{X}_{\Phi c}^{\mathrm{T}}$. Define the centered kernel matrix as $\mathbf{K}_c = \mathbf{X}_{\Phi c}^{\mathrm{T}}\mathbf{X}_{\Phi c}$. $\hat{\mathbf{K}}_{b\Phi}$ and $\mathbf{K}_c$ are real symmetric matrices, and they can be represented by its spectral decomposition as given by:

$$\hat{\mathbf{K}}_{b\Phi} = \mathbf{V}_{\Phi}\mathbf{\Lambda}_{\Phi}\mathbf{V}_{\Phi}^{\mathrm{T}} \tag{6}$$

$$\mathbf{K}_c = \mathbf{\Lambda}_c{}^{\mathrm{T}} \tag{7}$$

where $\mathbf{V}_{\Phi}$ and $\mathbf{A}$ are matrices whose columns are the eigenvectors, $\mathbf{\Lambda}_{\Phi}$ and $\mathbf{\Lambda}_{\mathrm{c}}$ are diagonal matrices of eigenvalues, respectively. Because $\mathbf{\Lambda}_{\mathrm{c}} = \mathrm{M}\mathbf{\Lambda}_{\Phi}$, $\mathbf{V}_{\Phi} = \mathbf{X}_{\Phi c}\mathbf{A}$ [20], the pseudo-inverse of the estimated background covariance matrix can be expressed as:

$$\hat{\mathbf{K}}_{b\Phi}^{\#} = \mathbf{V}_{\Phi}\mathbf{\Lambda}_{\Phi}^{-1}\mathbf{V}_{\Phi}^{\mathrm{T}} = \mathrm{M}\mathbf{X}_{\Phi c}\mathbf{K}_c^{-1}\mathbf{X}_{\Phi c}^{\mathrm{T}} \tag{8}$$

where M is a constant, and it can be ignored. Inserting Equation (8) into Equation (5) it can be rewritten as:

$$\begin{aligned}\mathrm{KRX}(\Phi(\mathbf{r})) &= (\Phi(\mathbf{r}) - \hat{\mu}_{b\Phi})^{\mathrm{T}}\hat{\mathbf{K}}_{b\Phi}^{-1}(\Phi(\mathbf{r}) - \hat{\mu}_{b\Phi}) \\ &= (\Phi(\mathbf{r}) - \hat{\mu}_{b\Phi})^{\mathrm{T}}\mathbf{X}_{\Phi c}\mathbf{K}_c^{-1}\mathbf{X}_{\Phi c}^{\mathrm{T}}(\Phi(\mathbf{r}) - \hat{\mu}_{b\Phi})\end{aligned} \tag{9}$$

Due to the high dimensionality of the feature space and the nonlinear mapping function $\Phi$ are unknown, it is difficult to compute directly in the feature space. Nevertheless, kernel-based learning method uses a kernel trick that the dot products in the feature space can be replaced by a kernel function [21], which is represented as:

$$\mathrm{k}(\mathrm{x}_i, \mathrm{x}_j) = < \Phi(\mathrm{x}_i), \Phi(\mathrm{x}_j) > = \Phi(\mathrm{x}_i) \cdot \Phi(\mathrm{x}_j) \tag{10}$$

Through kernel trick and derivation, then:

$$\begin{aligned} \Phi(\mathbf{r})^{\mathrm{T}} \mathbf{X}_{\Phi c} = & \Phi(\mathbf{r})^{\mathrm{T}} ([\Phi(\mathrm{x}_1), \Phi(\mathrm{x}_2), \cdots, \Phi(\mathrm{x}_M)] - \hat{\mu}_{b\Phi}) \\ & = \mathrm{k}(\mathbf{r}, \mathbf{X}_b) - \tfrac{1}{M} \sum_{i=1}^{M} \mathrm{k}(\mathbf{r}, \mathrm{x}_i) \mathbf{1}_{1\times M} \\ & \equiv \mathrm{k}_r^{\mathrm{T}} \end{aligned} \tag{11}$$

$$\begin{aligned} \hat{\mu}_{b\Phi}^{\mathrm{T}} X_{\Phi c} = & \hat{\mu}_{b\Phi}^{\mathrm{T}} ([\Phi(\mathrm{x}_1), \Phi(\mathrm{x}_2), \cdots, \Phi(\mathrm{x}_M)] - \hat{\mu}_{b\Phi}) \\ & = [\tfrac{1}{M} \sum_{i=1}^{M} \mathrm{k}(\mathrm{x}_i, \mathbf{X}_b)] - [\tfrac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} \mathrm{k}(\mathrm{x}_i, \mathrm{x}_j)] \mathbf{1}_{1\times M} \\ & \equiv \mathbf{k}_{\hat{\mu}}^{\mathrm{T}} \end{aligned} \tag{12}$$

where $\mathbf{1}_{1\times \mathrm{M}}$ denotes $M$ dimensional row vector whose elements are all 1. Due to the centralization of the feature space cannot be obtained, it calculates $\mathbf{K}_c$ via $\mathbf{K}\mathrm{b} = \Phi(\mathbf{X}_b)^{\mathrm{T}} \Phi(\mathbf{X}_b)$:

$$\mathbf{K}_c = \mathbf{K}_b - \mathbf{K}_b \mathbf{I}_M - \mathbf{I}_M \mathbf{K}_b + \mathbf{I}_M \mathbf{K}_b \mathbf{I}_M \tag{13}$$

where $\mathbf{I}_M$ is an M-dimensional square matrix of all elements as 1/M. Finally, the KRX algorithm can be simplified as:

$$\mathrm{KRX}(\Phi(\mathbf{r})) = (\mathbf{k}_r^{\mathrm{T}} - \mathbf{k}_{\hat{\mu}}^{\mathrm{T}})^{\mathrm{T}} \mathbf{K}_c^{-1} (\mathbf{k}_r^{\mathrm{T}} - \mathbf{k}_{\hat{\mu}}^{\mathrm{T}}) \tag{14}$$

## *2.2. WSSKRX Algorithm and Parallel Implementation on GPUs*

The original KRX algorithm makes full use of the nonlinear information between bands by kernel mapping. Unfortunately, when the anomaly information is mixed into the background data, the kernel mapping cannot represent the ideal distribution of the background data, and it affects the detection performance. Furthermore, one limitation of KRX algorithm is that it has highly computational complexity because of massive nonlinear kernel function and inverse of covariance matrices. Accordingly, the novel WSSKRX detector is proposed in this paper, and a parallel architecture on GPUs is designed according to the characteristics of WSSKRX.

### 2.2.1. The Local Information Reconstruction

For any pixel $\mathbf{r}_i = (r_{i1}, r_{i2}, \ldots, r_{iL})^{\mathrm{T}}$ where $L$ is the total number of spectral bands, its neighborhood information is spectral correlative and spatial correlative. Assume that $\Omega(\mathbf{r}_i)$ is a local sliding window with $\mathbf{r}_i$ being the centered pixel and the size of window being $w^2 = w \times w$ where $w$ is an odd number and positive integer (as shown in Figure 2). The representation form of $\Omega(\mathbf{r}_i)$ is given by:

$$\Omega(\mathbf{r}_i) = \{\mathbf{r}_p | p \in [i-a, i+a]\} \tag{15}$$

where $\mathbf{r}_p$ is a pixel within the neighborhood window, and $a = (w^2 - 1)/2$ is a constant. In order to better capture the spatial information, the centered pixel is reconstructed according to the weighted spatial-spectral information in this paper. The reconstructed pixel $\hat{\mathbf{r}}_i$ is specified by:

$$\hat{\mathbf{r}}_i = \frac{\sum_{\mathbf{r}_p \in \Omega(\mathbf{r}_i)} \omega_p \mathbf{r}_p}{\sum_{\mathbf{r}_p \in \Omega(\mathbf{r}_i)} \omega_p} \tag{16}$$

where $\omega_p = \exp(-t||\mathbf{r}_i - \mathbf{r}_p||_2^2)$ is the weight of any pixel $\mathbf{r}_p$ to center pixel $\mathbf{r}_i$ in the spatial neighborhood $\Omega(\mathbf{r}_i)$, $||\bullet||_2$ denotes two norm operation. $t > 0$ is a spectral factor, indicating the degree of interaction effects between different pixels in the same neighboring space. Accordingly, we can get the kernel function between the reconstructed pixels as:

$$\begin{aligned} \mathrm{k}_{SS}(\hat{\mathbf{r}}_i, \hat{\mathbf{r}}_j) &= < \Phi(\hat{\mathbf{r}}_i), \Phi(\hat{\mathbf{r}}_j) > \\ &= \left\langle \frac{\sum_{\mathbf{r}_p \in \Omega(\mathbf{r}_i)} \omega_p \mathbf{r}_p}{\sum_{\mathbf{r}_p \in \Omega(\mathbf{r}_i)} \omega_p}, \frac{\sum_{\mathbf{r}_q \in \Omega(\mathbf{r}_j)} \omega_q \mathbf{r}_q}{\sum_{\mathbf{r}_q \in \Omega(\mathbf{r}_j)} \omega_q} \right\rangle \\ &= \frac{\sum_{\mathbf{r}_p \in \Omega(\mathbf{r}_i)} \sum_{\mathbf{r}_q \in \Omega(\mathbf{r}_j)} \omega_p \omega_q \mathrm{k}(\mathbf{r}_p, \mathbf{r}_q)}{\sum_{\mathbf{r}_p \in \Omega(\mathbf{r}_i)} \omega_p \sum_{\mathbf{r}_q \in \Omega(\mathbf{r}_j)} \omega_q} \end{aligned} \quad (17)$$

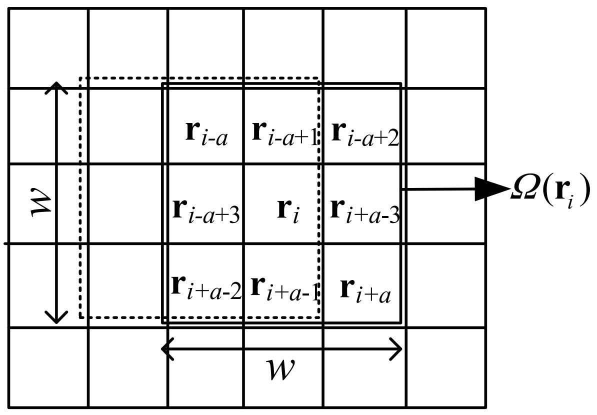

$\mathbf{r}_{i\text{-}a}$
$\mathbf{r}_{i\text{-}a+1}$
$\mathbf{r}_{i\text{-}a+2}$
$\mathbf{r}_{i\text{-}a+3}$
$\mathbf{r}_i$
$\mathbf{r}_{i+a\text{-}3}$
$\mathbf{r}_{i+a\text{-}2}$
$\mathbf{r}_{i+a\text{-}1}$
$\mathbf{r}_{i+a}$
$w$
$w$
$\Omega(\mathbf{r}_i)$

**Figure 2.** The local sliding window $\Omega(\mathbf{r}_i)$.

The weight $\omega_p$ is greater when the spectral curve of two pixels are closer and $\omega_p$ is determined by the spectral factor $t$. Furthermore, a pixel will be incorporated into the center pixel with a smaller weight when it is distinct greatly from the center pixel in the neighborhood space. For example, if the PUT is the background pixel and anomalies exist in the background sample data, the reconstructed PUT will be closer to the background features and weaken the influence of abnormal information on background data. Therefore, the local PUT reconstruction can effectively avoid performance degradation duo to background data are contaminated by anomalous information. It should be noted that the reconstruction and detection of the edge pixels need to rely on the edge expansion of HSI. The sliding window is to achieve the reconstruction task of each pixel as shown in Figure 3.

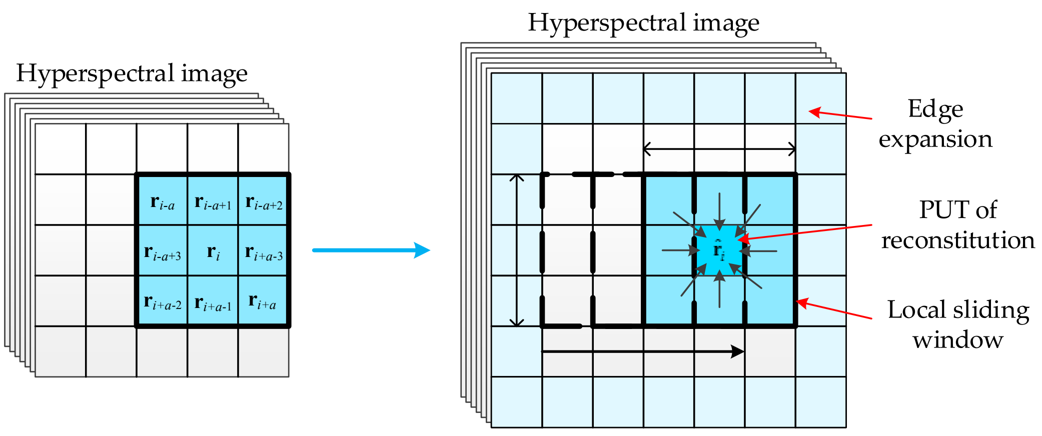

Hyperspectral image
$\mathbf{r}_{i\text{-}a}$
$\mathbf{r}_{i\text{-}a+1}$
$\mathbf{r}_{i\text{-}a+2}$
$\mathbf{r}_{i\text{-}a+3}$
$\mathbf{r}_i$
$\mathbf{r}_{i+a\text{-}3}$
$\mathbf{r}_{i+a\text{-}2}$
$\mathbf{r}_{i+a\text{-}1}$
$\mathbf{r}_{i+a}$
Hyperspectral image
$\hat{\mathbf{r}}_i$
Edge expansion
PUT of reconstitution
Local sliding window

**Figure 3.** Illustration of information reconstitution.

Window size $w$ and spectral factor $t$ are two important parameters of information reconstruction. Generally, the selection of the parameter $w$ and $t$ depend on the image itself. When the object distribution is more concentrated, the spatial relationship between the pixels is more obvious, and the spectral difference is smaller. So we should choose larger $w$ and smaller $t$. On the contrary, when the distribution is more dispersed, we need to choose a smaller $w$ to represent weaker spatial relationship and larger $t$ to weaken the impact of abnormal information.

### 2.2.2. Weighted Spatial-Spectral Kernel RX Method

The detection performance of kernel-based method depends on the form of kernel function and the selection of kernel parameter. The effective use of spatial-spectral information in kernel-based anomaly detection is helpful to optimize the kernel mapping in high-dimensional feature space, thereby improving the performance of anomaly detection. We redefine the feature vector of the sample point $\widetilde{\mathbf{r}}_i$ which includes spatial information $\hat{\mathbf{r}}_i$ and spectral information $\mathbf{r}_i$. When the spatial and the spectral vector are constructed, the kernel function can be obtained by satisfying the kernel function of the Mercer condition [22], which is specified by:

$$\begin{cases} \mathrm{k}(\widetilde{\mathbf{r}}_i,\widetilde{\mathbf{r}}_j) = \mu \mathrm{k}_1(\hat{\mathbf{r}}_i,\hat{\mathbf{r}}_j) + (1-\mu)\mathrm{k}_2(\mathbf{r}_i,\mathbf{r}_j) \\ \text{s.t. } 0 \le \mu \le 1 \end{cases} \tag{18}$$

where $\mu$ is a weighting factor. Gaussian radial basis function (RBF) kernel has a translation invariance and better capability of local information retrieval in the feature space. Therefore, in this paper, we use RBF kernel as the base kernel mapping function:

$$\mathrm{k}(\mathbf{x},\mathbf{y}) = \exp(-||\mathbf{x}-\mathbf{y}||_2/c) \tag{19}$$

where $c$ is a positive constant which represents the width of the RBF kernel. According to Equations (17) and (18), $\Phi(\mathbf{r})^{\mathrm{T}}\mathbf{X}_{\Phi c}$ in Equation (11) can be re-expressed as:

$$\begin{aligned} \Phi(\widetilde{\mathbf{r}})^{\mathrm{T}}\mathbf{X}_{\Phi c} &= \Phi(\widetilde{\mathbf{r}})^{\mathrm{T}}([\Phi(\mathrm{x}_1),\Phi(\mathrm{x}_2),\cdots,\Phi(\mathrm{x}_M)] - \hat{\boldsymbol{\mu}}_{b\Phi}) \\ &= \mathrm{k}(\widetilde{\mathbf{r}},\mathbf{X}_b) - \tfrac{1}{M}\sum_{i=1}^{M}\mathrm{k}(\widetilde{\mathbf{r}},\mathrm{x}_i)\mathrm{I}_{1\times M} \\ &= \boldsymbol{\mu}\mathrm{k}(\hat{\mathbf{r}},\mathbf{X}_b) + (1-\mu)\mathrm{k}(\mathbf{r},\mathbf{X}_b) - \tfrac{1}{M}\sum_{i=1}^{M}[\mu\mathrm{k}(\hat{\mathbf{r}},\mathrm{x}_i) + (1-\mu)\mathrm{k}(\mathbf{r},\mathrm{x}_i)]\mathrm{I}_{1\times M} \\ &= \mu\left[\mathrm{k}(\hat{\mathbf{r}},\mathbf{X}_b) - \tfrac{1}{M}\sum_{i=1}^{M}\mathrm{k}(\hat{\mathbf{r}},\mathrm{x}_i)\mathrm{I}_{1\times M}\right] + (1-\mu)\left[\mathrm{k}(\mathbf{r},\mathbf{X}_b) - \tfrac{1}{M}\sum_{i=1}^{M}\mathrm{k}(\mathbf{r},\mathrm{x}_i)\mathrm{I}_{1\times M}\right] \\ &= \mu\Phi(\hat{\mathbf{r}})^{\mathrm{T}}\mathbf{X}_{\Phi c} + (1-\mu)\Phi(\mathbf{r})^{\mathrm{T}}\mathbf{X}_{\Phi c} \\ &\equiv \widetilde{\mathrm{k}}_r^{\mathrm{T}} \end{aligned} \tag{20}$$

where $0 \le \mu \le 1$, $\hat{\mathbf{r}}$ is the reconstructed sample pixel of PUT, and $\mathbf{r}$ is the spectral information of PUT. Therefore, we can re-write the expression of KRX as:

$$\mathrm{WSSKRX}(\Phi(\widetilde{\mathbf{r}})) = (\widetilde{\mathbf{k}}_r^{\mathrm{T}} - \mathbf{k}_{\hat{\boldsymbol{\mu}}}^{\mathrm{T}})^{\mathrm{T}}\mathbf{K}_c^{-1}(\widetilde{\mathbf{k}}_r^{\mathrm{T}} - \mathbf{k}_{\hat{\boldsymbol{\mu}}}^{\mathrm{T}}) \tag{21}$$

As illustrated in Figure 4, the WSSKRX algorithm utilizes a local dual concentric windows model to detect anomalies. The inner window is used to avoid the potential target information falling into the background. And PUT reconstruction, the local kernel matrix and background covariance matrix are calculated from the pixel vectors in the outer window.

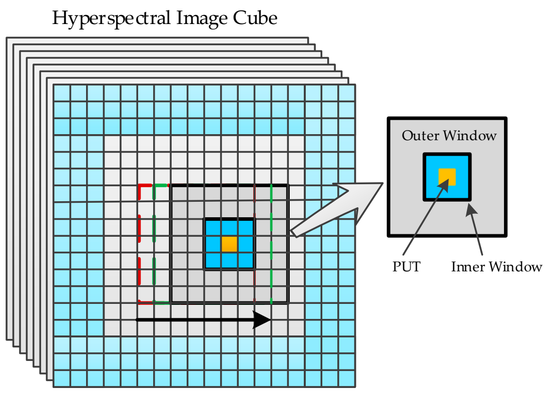


**Figure 4.** Dual concentric windows model for WSSKRX anomaly detection.

### 2.2.3. Parallel Implementation of the WSSKRX Algorithm on GPUs

A GPU is actually an array of streaming multiprocessors (SMs) in which each multiprocessor is featured by a single instruction multiple data (SIMD) architecture in each clock cycle. Each processor executes the same instruction but operates on multiple data streams. Significantly, there are multiple streaming processors (SPs) in each SM, which could be considered as many CPU cores. CPUs often run single-thread programs, and they only calculate a single data point per core, per iteration, while GPUs run in parallel by default. Thus, instead of calculating a single data point per SM, GPUs calculate 32 per SM. This gives a 32 times advantage in terms of data throughput.

In Figure 5, "Host" represents the computer and "Device" represents GPUs. In device, there are different levels of memory. Thread private data will be assigned to the so-called local memory, when an excess of registers are used or the registers are depleted. In the computation, we would rather take full advantage of shared memory, which can be shared to threads in the same block to write/read quickly. The Global memory which could provide a wide memory bandwidth is supplied via Graphic Double Data Rate (GDDR) on the graphics card. It is a high-performance version of Double Data Rate (DDR) memory.

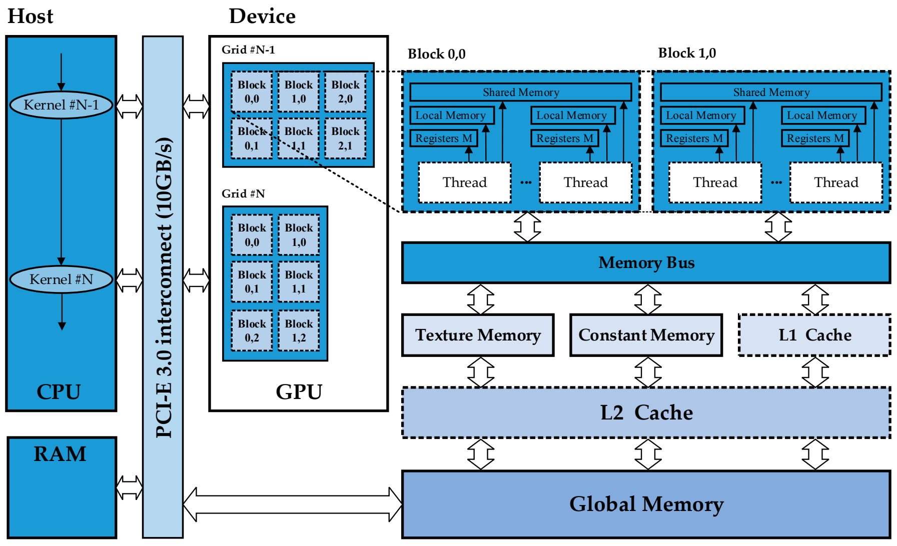


**Figure 5.** The architecture and operation of NVIDIA GPU, and the data transfer between CPU and GPU.

CUDA takes a simple model of data parallelism and incorporates it into a programming model without the need for graphics primitives. Being the easiest language to develop, CUDA has a huge lead in terms of maturity. CUDA was launched by NVIDIA as a kind of general computing architecture, which can enable GPU to work out many complicated calculations. In the process of a calculation, CUDA uses a grid of blocks. This can be thought as a queue (or a grid) of processes (blocks) with no inter-process communication. Each block has many threads which operate cooperatively in batches called warps.

In this paper, the parallel version of the WSSKRX on GPU for anomaly detection is given by following steps (Algorithm 1). It is noted that the underlined parts are parallel computation executed on GPU.

**Algorithm 1.** CUDA Pseudo-Code of WSSKRX

```
1: INPUTS: D, N, L ,$w_{out}^2$, $w_{in}^2$, $t$, $c$ and $\mu$
//D denotes a hyperspectral image with N pixel, L denotes the number of spectral bands, $w_{out}^2$ and $w_{in}^2$ denote the size of outer and inner window, respectively, $t$ denotes spectral factor, $c$ denotes the width of the RBF kernel, and $\mu$ denotes weighting factor.
2: cudaMemcpy ();
//copy the initial hyperspectral data from host to device (data communication)
3: $\underline{\text{the edge expansion of } \mathbf{D};}$
//the size of expansion is determined by the $w_{out}^2$
4: $\mathbf{r}_i = (r_{i1}, r_{i2}, \cdots, r_{iL})^T, \mathbf{r}_i \in \mathbf{D}$;
//$\mathbf{r}_i$ denotes $i^{th}$ data sample vector
5: For (N)
6: $\underline{\text{Calculate } \hat{\mathbf{r}}_i = (\hat{r}_{i1}, \hat{r}_{i2}, \cdots, \hat{r}_{iL})^T \text{ in } w_{out}^2;}$
//$\hat{\mathbf{r}}_i$ denotes reconstructed information of sample vector
7: cudaThreadSynchronize ();
//the function of thread synchronization
8: $\underline{\text{Calculate } \widetilde{\mathbf{k}}_r^{\mathrm{T}} \text{ and } \mathbf{k}_{\hat{\mu}}^{\mathrm{T}};}$
//$\widetilde{\mathrm{k}}_r^{\mathrm{T}}$ and $\mathbf{k}_{\hat{\mu}}^{\mathrm{T}}$ denoted the kernel operations in the feature space
9: cudaThreadSynchronize ();
//the function of thread synchronization
10: $\underline{\text{Calucalate } \mathbf{K}_c = \mathbf{K}_b - \mathbf{K}_b * \mathbf{I}_M - \mathbf{I}_M * \mathbf{K}_b + \mathbf{I}_M * \mathbf{K}_b * \mathbf{I}_M;}$
//$\mathbf{K}_c$ denoted Gram matrix, $\mathbf{K}_b = \Phi(\mathbf{X}_b)^{\mathrm{T}} * \Phi(\mathbf{X}_b)$
11: $\underline{\mathbf{K}_c^{-1} = \mathrm{inv}(\mathbf{K}_c);}$
12: $\underline{\text{result} = (\widetilde{\mathbf{k}}_r^{\mathrm{T}} - \mathbf{k}_{\hat{\mu}}^{\mathrm{T}})^{\mathrm{T}} * \mathbf{K}_c^{-1} * (\widetilde{\mathbf{k}}_r^{\mathrm{T}} - \mathbf{k}_{\hat{\mu}}^{\mathrm{T}});}$
13: cudaMemcpy();
//copy the hyperspectral data from device to host (data communication)
14: End for
15: OUTPUT: result
```

We take one or more streams as inputs and produce one or more streams as outputs by a multiprocessor. On the host, the CPU executes the kernel in CUDA, then the grid dimension is determined on the device side according to the size of matrices and the number of stream processors. Through the design of the CUDA kernel, a certain parallel task can be accomplished. Each thread will have a clear positioning by the rational allocation of thread resources, so as to achieve uniform scheduling of threads. In this implementation, we design multiple CUDA kernels, which can complete different function respectively. The following mainly introduce the four categories of CUDA kernels:

(1) In order to facilitate the detection of edge information, the original data need to be expanded. Thus, the first CUDA kernel (step 3) is designed to copy the original edge information to new extended boundaries. *Row = blockIdx.x * blockDim.x + threadIdx.x* and *Col = blockIDx.y * blockDim.y + threaddx.y* have been used to achieve the index of thread and useful threads have been systematically scheduled. Thereby realizing the parallel process of information expansion.

(2) The second CUDA kernel (step 6) fetches the PUTs from dual concentric windows and achieves the process of information reconstruction. The kernel launches as many blocks as the number of pixels for the original data presented in step 2, where each thread in one block computes an element of $\hat{\mathbf{r}}_i = (\hat{r}_{i1}, \hat{r}_{i2}, \cdots, \hat{r}_{iL})^T$ and then stores their reconstructed information into the global memory.

(3) The kernel function takes a certain amount of computation in the algorithm and then the third kind of CUDA kernel is designed for each kernel operation in global memory. All the computational tasks are allocated to multiple threads for independent parallel implementation (steps 8–11). The CUDA kernel uses the sub-block of matrix and multi-warps independent operation to reduce memory latency and processing time. Since the calculation of the centralized Gram matrix $\mathbf{K}_c$ is related to $\mathbf{k}_{\hat{\mu}}^{\mathrm{T}}$, a barrier should be created before calculation of $\mathbf{K}_c$ to ensure correctness of the calculation. Through a series of parallel optimization design, the computational complexity of kernel function and matrix inversion are effectively reduced.

(4) And the last CUDA kernel (step 12) computes the results and completes target detection. To minimize the number of the global memory accesses and reduce access time, the result vector of $(\mathbf{k}_r^{\mathrm{T}} - \mathbf{k}_{\mu}^{\mathrm{T}})^T$ and matrix $\mathbf{K}_c^{-1}$ are partitioned into sub-blocks and transferred to the shared memory. Each block uses a total of 48 KB of shared memory. Figure 6 illustrates this procedure, where each thread is responsible for computing each element of $(\mathbf{k}_r^{\mathrm{T}} - \mathbf{k}_{\mu}^{\mathrm{T}})^{\mathrm{T}} * \mathbf{K}_c^{-1}$. For every matrix multiplication, each thread in one warp is responsible for one calculation, in which a row of the matrix is multiplied by a column. For two vector multiplication, every corresponding element of them are multiplied and added in turn. It takes plenty of time and threads. One common approach to solve this problem is parallel reduction. It works by using half number of threads of the elements in the dataset. Every thread calculates the sum of products. The resultant element is forwarded to the next round. In addition, the number of threads is then reduced by half and the process repeats until there is just a single element remaining (in Figure 7). A better approach is to drop whole warps by selecting the element from the other half of the dataset.

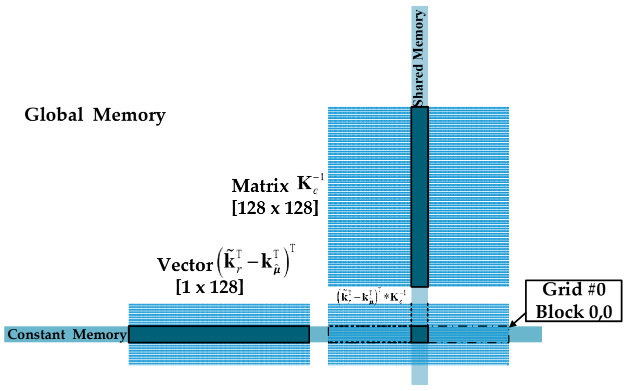

Shared Memory
Global Memory
Matrix $\mathbf{K}_c^{-1}$
[128 x 128]
Vector $\left(\tilde{\mathbf{k}}_r^{\mathrm{T}} - \mathbf{k}_{\hat{\mu}}^{\mathrm{T}}\right)^{\mathrm{T}}$
[1 x 128]
$\left(\tilde{\mathbf{k}}_r^{\mathrm{T}} - \mathbf{k}_{\hat{\mu}}^{\mathrm{T}}\right)^{\mathrm{T}} * \mathbf{K}_c^{-1}$
Grid #0
Block 0,0
Constant Memory

**Figure 6.** Illustration of thread for computing one element of result in GPU.

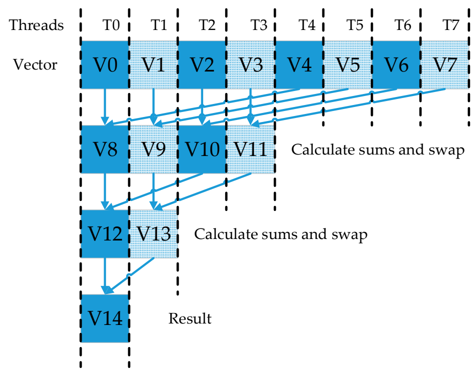


**Figure 7.** Reduction process in CUDA.

## 3. Description of Hyperspectral Datasets

### *3.1. Synthetic Dataset*

In Figure 8a, The HyMap image was acquired by HyMap hyperspectral remote sensor in the Cook City, MT, USA. There are 126 bands ranging from 0.4 to 2.5 µm with a size of 280 × 800 pixels. The background of synthetic dataset is set as the real ground which cut out the size of 90 × 90 pixels from the HyMap image (white box in Figure 8a). And the targets of synthetic dataset are designed by an airborne hyperspectral image collected by the AVIRIS imaging sensor from San Diego airport. The four targeted spectral signatures (G, H, T, P), marked by circles in Figure 8b, are used to form the synthetic targets. The four spectral signatures are used to simulate 16 targets shown in Figure 8c with 4 targets in each row simulated by same spectral signature. The sizes of panels from left column to right column are 4 × 4, 3 × 3, 2 × 2, 1 × 1, respectively. The panels in the first column are truncated from the original image, and others in 2–4 column for each row are superposed by a different proportion of background interference. And the ground truth of synthetic dataset is given by Figure 8d.

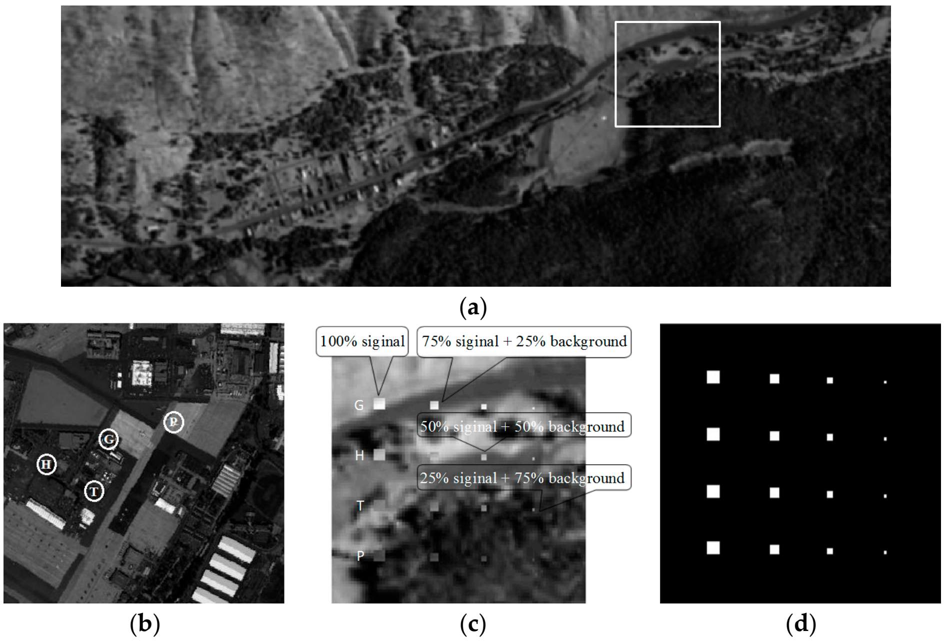

(a)
G
H
T
P
100% siginal
75% siginal + 25% background
50% siginal + 50% background
25% siginal + 75% background
(b)
(c)
(d)

**Figure 8.** (**a**) HyMap hyperspectral image; (**b**) SanDiego Airport image; (**c**) synthetic image; (**d**) ground truth of synthetic image.

### *3.2. SanDiego Airport Dataset*

The San Diego Airport dataset, collected by the AVIRIS hyperspectral spectrometer, covers the area of San Diego airport. The original data contains 400 × 400 pixels with a 3.5 m space resolution and 224 bands with a 10 nm spectral resolution (in Figure 8b).

In this study, one subarea which contains 60 × 60 pixels and 126 bands is selected for experiment. In Figure 9a,b show the subarea of airport image and the ground truth of image scene, respectively.

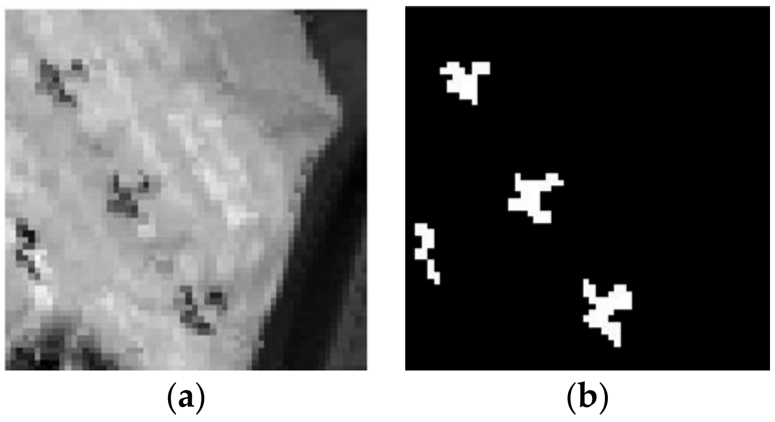


**Figure 9.** (**a**) The subarea of airport image; (**b**) ground truth of the airport image scene.

### *3.3. SpecTIR Dataset*

The dataset were collected through the SpecTIR Hyperspectral Airborne Rochester Experiment (SHARE) by the ProSpecTIR-VS2 sensor (SpecTIR, LLC, Rochester, New York, NY, USA). It contains 3127 × 320 pixels with an 1 m space resolution, and 360 bands ranging from 390 nm to 2450 nm with a 5 nm spectral resolution. In this experiment, a subset with a size of 100 × 100 pixels and 360 bands is segmented from the original data. There are some square fabrics placed as anomaly targets in the subarea of image (Figure 10a), and the ground truth map is shown in Figure 10b.

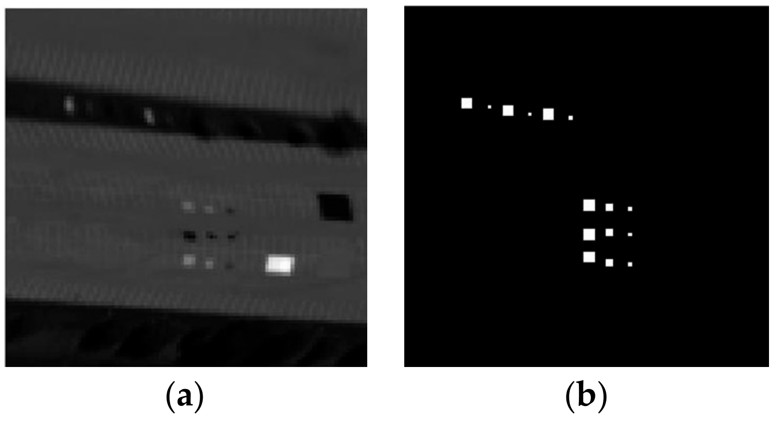


**Figure 10.** (**a**) The subarea of SpecTIR image; (**b**) ground truth of the SpecTIR image scene.

## 4. Experimental Results and Analysis

In this section, to investigate the anomaly detection accuracy and parallel performance, experiments are conducted one synthetic dataset and two real datasets. All the experiments are performed on a 2.0 GHz Intel Xeon E7-4820, running on a 64-bits operating system with 96 GB of RAM memory. The parallel version is implemented in CUDA C programing language for GPU cards of NVIDIA Tesla K40m. WSSKRX algorithm is compared with Global RX (GRX), Local RX (LRX), Local KRX (LKRX) and Hybrid Kernel RX (HKRX) [23] to evaluate the detection accuracy. HKRX algorithm extract the global and local feature information effectively via adding a modified spectral angle kernel to Gaussian kernel function to improve the detection performance. Other competitors have been introduced in Section 2.

### *4.1. Effects from the Parameter on WSSKRX*

For the parameters of the algorithm, by virtue of 4-fold cross-validation, the RBF kenrel parameter $c$ in WSSKRX on synthetic dataset, San Diego Airport dataset and SpecTIR dataset is 5, 2 and 2, respectively. The spectral factor $t$ is set to 2, 2 and 10, and the weighting factor $\mu$ of WSSKRX is set to be 0.6, 0.5 and 0.4 on synthetic dataset, San Diego Airport dataset and SpecTIR dataset, respectively. The window size ($w_{in}$,$w_{out}$) is set to (3,11), (5,11) and (3,11) on three datasets, respectively.

Figure 11 shows the different AUC of WSSKRX with changing spectral factor $t$ on three hyperspectral images. It is clear that if spectral factor $t$ is less than 1 or larger than 120, the AUC descends immediately for the San Diego Airport dataset, and the corresponding AUC is the highest when $t = 2$. For SpecTIR dataset and synthetic dataset, the AUCs tend to be stable when the spectral factor is between 1 and 30, and they reach a maximum at $t = 10$ and $t = 2$, respectively. Therefore, it is concluded that spectral factor $t$ is not sensitive in some numerical range. Generally, when the distribution of data features are more concentrated, the selected $t$ should be smaller.

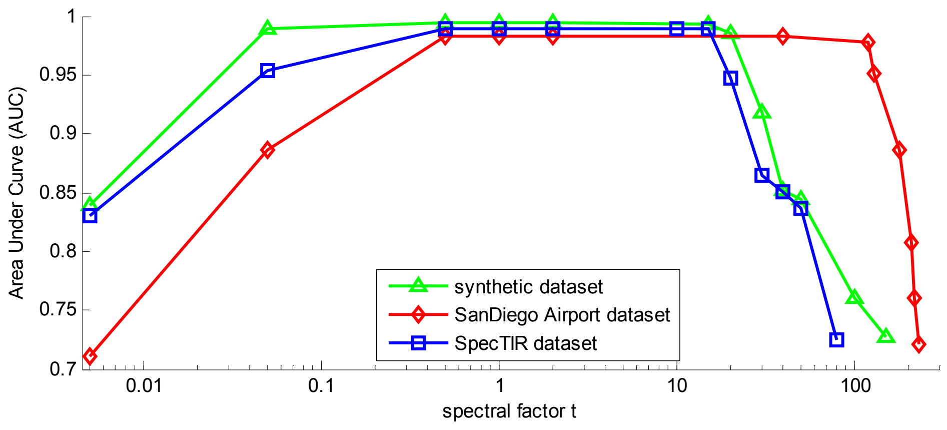
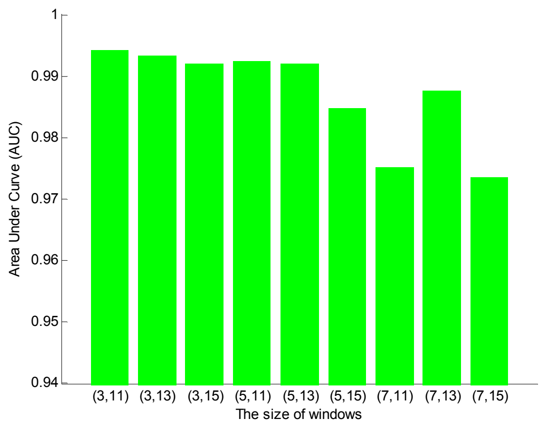
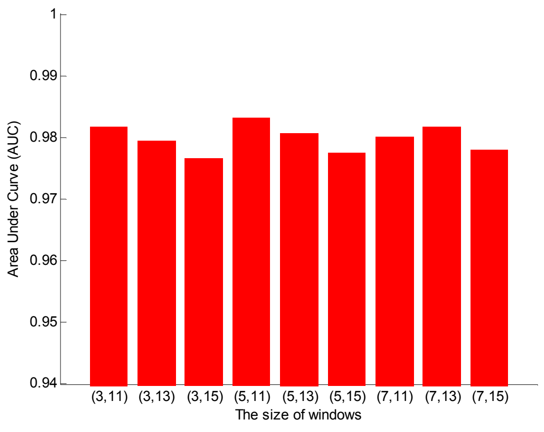


**Figure 11.** The AUC of WSSKRX with the changing spectral factor $t$ on the San Diego Airport dataset, SpecTIR dataset and synthetic dataset.

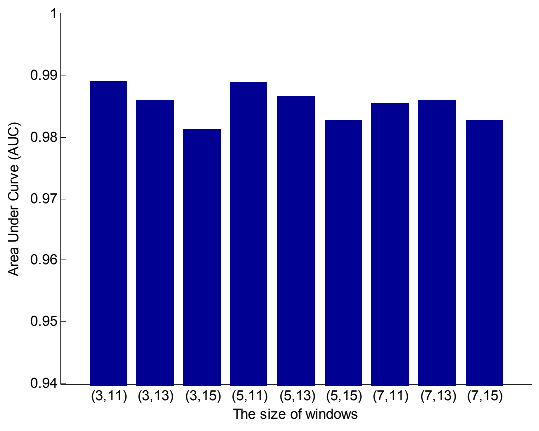
Figure 12 gives the AUC curves of WSSKRX using different size of dual windows on three datasets, respectively. When the size of windows on synthetic dataset, San Diego Airport dataset and SpecTIR dataset are (3, 11), (5, 11) and (3, 11) respectively, the detection performance of WSSKRX is optimal. The optimal inner window size $w_{in}$ used to protect anomaly information is also larger, as the anomaly target of the San Diego Airport dataset is larger than other two datasets. The window sizes in general depend on the size of target pixel in the image and local-double-window model is more suitable for small target data processing. Generally, the analysis of window sizes were considered in the range from (3, 11) to (7, 15) in the field of anomaly detection. For the RBF kernel parameter and the weighting factor, we can determine a proper parameter size by cross-validation.



(**a**)



(**b**)



(**c**)

**Figure 12.** The AUC of WSSKRX with the changing window size $(w_{in}, w_{out})$. (**a**) Synthetic dataset; (**b**) San Diego Airport dataset; (**c**) SpecTIR dataset.

## 4.2. Detection Accuracy of WSSKRX

In the experiments, for WSSKRX, all of parameters are the same as in Section 4.1. For the methods used as comparison, cross-validation is used to get optimal parameters. For LKRX and HKRX, the parameter $c$ of RBF kernel is set to be 5 on synthetic dataset, 2 on two real datasets. For HKRX, the parameter $d$ of spectral angle kernel is set to be 2 on synthetic and San Diego Airport dataset, 4 on SpecTIR dataset and the weighting factor $\alpha$ is set to be 0.7, 0.4 and 0.5 on three datasets, respectively. The size of dual windows in LKRX, HKRX and LRX, is the same as WSSKRX on three datasets.

Figures 13 and 14 show the grayscale and three-dimensional plots outputs of GRX, LRX, LKRX, HKRX and WSSKRX on three hyperspectral images, respectively. In Figure 13, LKRX, HKRX and WSSKRX can detect more abnormal pixels compared with GRX and LRX. This is because LKRX, HKRX and WSSKRX which are based on the kernel function exploit the nonlinear characteristics between spectral bands. Simultaneously, as WSSKRX algorithm effectively combines the spatial-spectral information, the detection accuracy is higher than other four detectors. In Figure 14, it can be seen that WSSKRX has better ability of suppressing noise interference compared with the LKRX and HKRX. From the observations, the separability performance of WSSKRX is better than other algorithms.

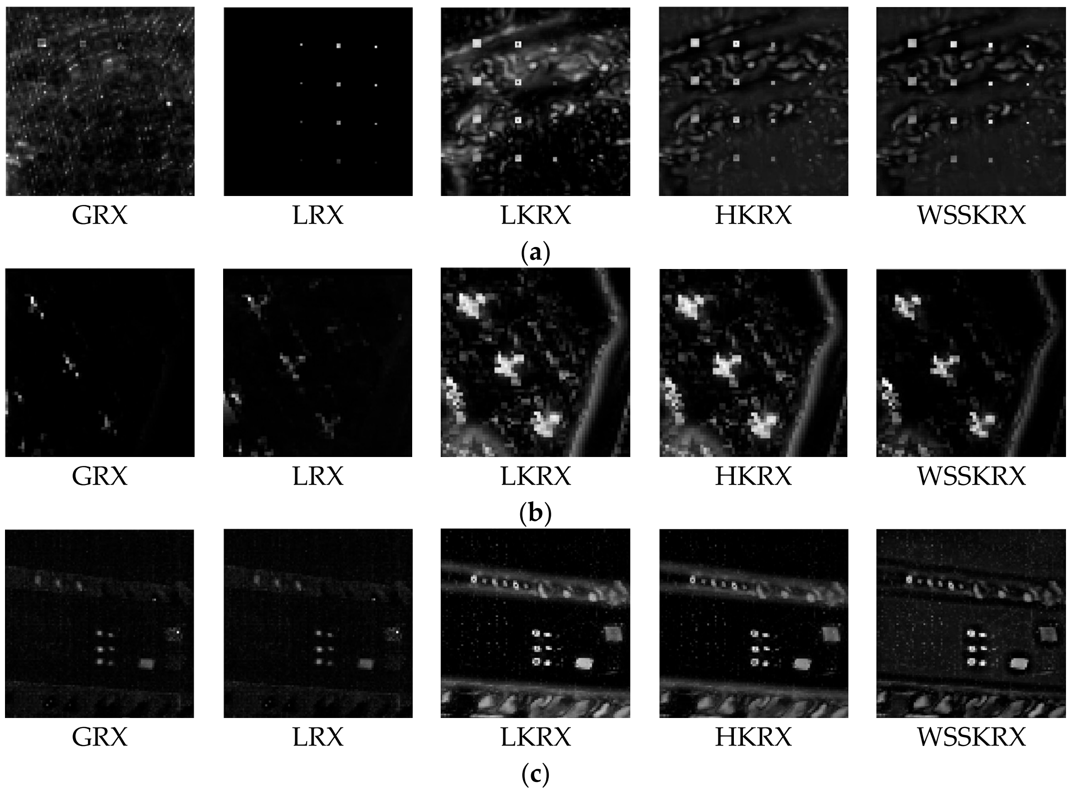


**Figure 13.** The grayscale detection results of GRX, LRX, LKRX, HKRX and WSSKRX on three datasets synthetic dataset San Diego Airport dataset SpecTIR dataset. (**a**) The grayscale of synthetic dataset; (**b**) The grayscale of San Diego Airport dataset; (**c**) The grayscale of SpecTIR dataset.

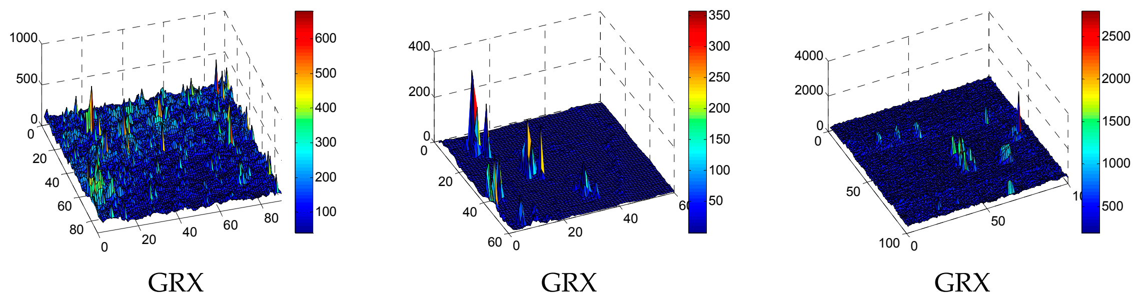


**Figure 14.** *Cont.*

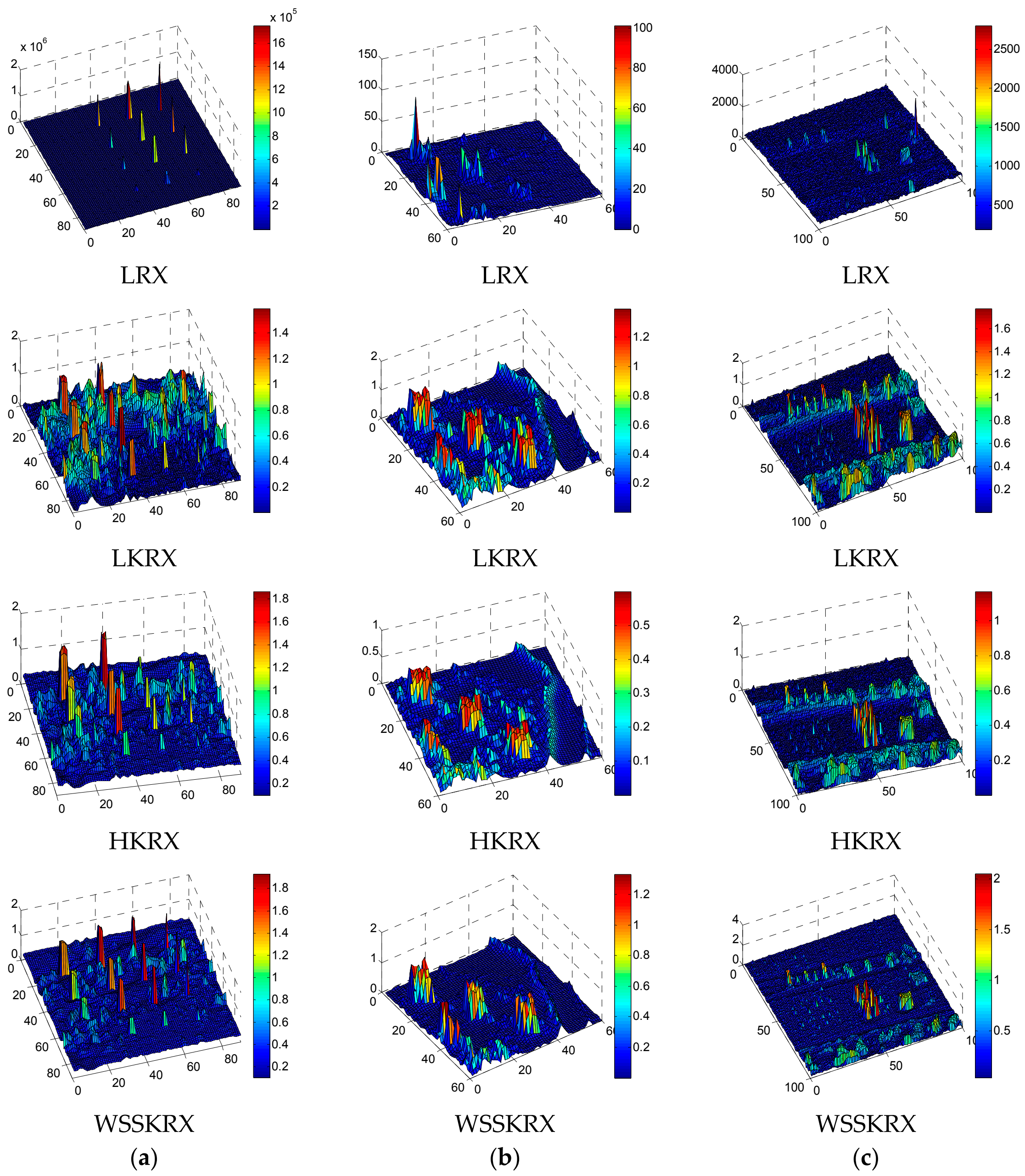


**Figure 14.** The 3D plots detection results of GRX, LRX, LKRX, HKRX and WSSKRX on three datasets. (**a**) The 3D plots of synthetic dataset; (**b**) The 3D plots of San Diego Airport dataset; (**c**) The 3D plots of SpecTIR dataset.

In order to give a more objective evaluation, Figure 15 presents the ROC curves of GRX, LRX, LKRX, HKRX and WSSKRX on three datasets. Since the feature distribution of the synthetic dataset is more complex and the GRX and LRX are operated in low-dimensional space, they perform worse as shown in Figure 15a. The detection accuracy of HKRX is better than KRX because the global and local information is taken into account together. The WSSKRX obtains better detection performance than other detectors, because it more rationally uses the data information. In Figure 15b, GRX and LRX perform worse than other algorithms. The ROC curves of LKRX, HKRX and WSSKRX are the similar when the false alarm rate is lower than 0.06. After that, WSSKRX begins to obtain a higher detection probability than LKRX and HKRX, and it achieves 1 probability of detection with a lower false alarm rate than LKRX and HKRX. In Figure 15c, Although WSSKRX needs a slightly higher false alarm rate than LKRX and HKRX when achieving 1 probability of detection, the overall detection performance of WSSKRX is still clearly better than other algorithms. It is noted that the feature distribution of the SpecTIR dataset is more separable than other two datasets, so GRX obtains better detection performance.

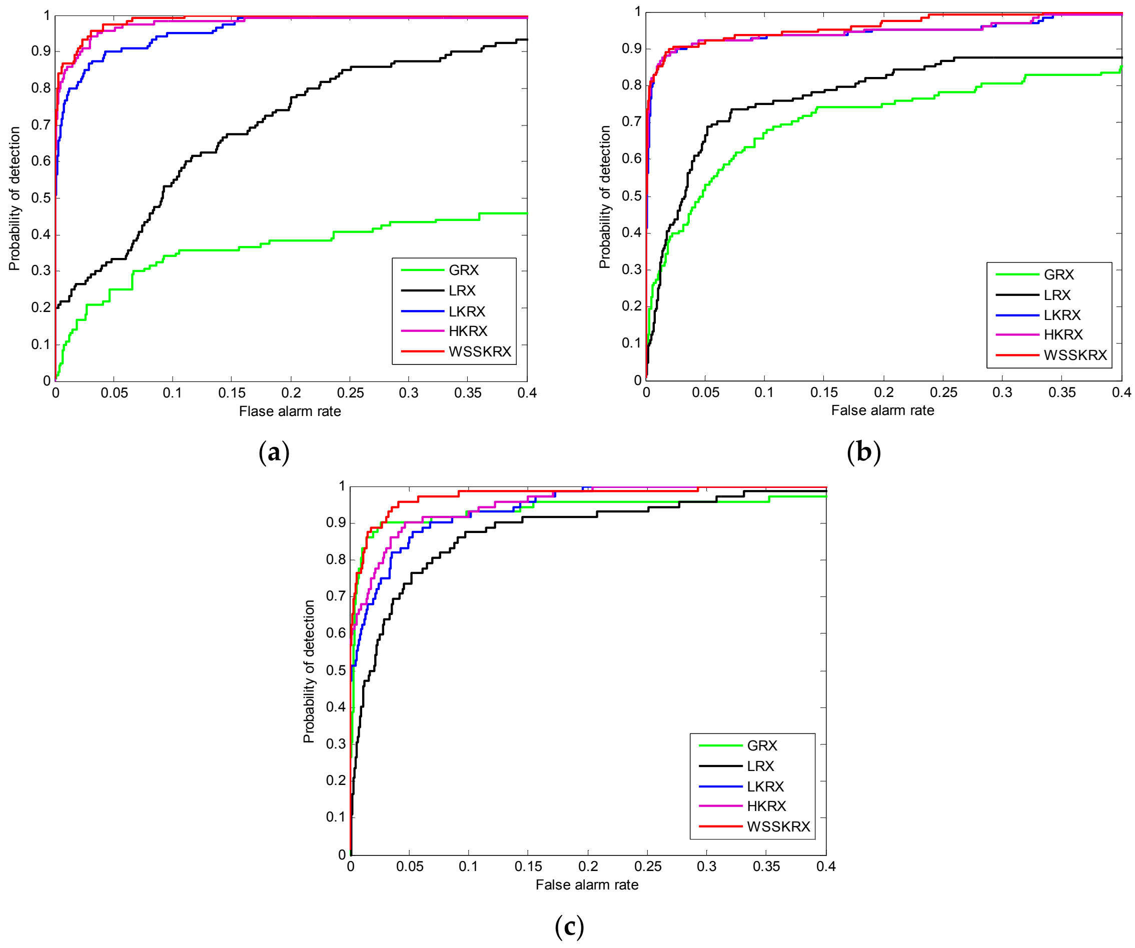


**Figure 15.** ROC curves of LRX, LKRX and WSSKRX on the two real datasets. (**a**) Synthetic dataset; (**b**) San Diego Airport dataset; (**c**) SpecTIR dataset.

## *4.3. Analysis of Parallel Performance*

In this subsection, the three datasets are used to validate the improved performance of GPU version when compared with CPU version. We evaluate the parallel performance of WSSKRX in NVIDIA Tesla K40m. Table 1 shows the processing times measured for the three hyperspectral images, along with computational speedup of the implementation on GPUs and CPU for WSSKRX. The C function "clock ()" was used for timing the CPU processing, and the CUDA timer was used for GPU implementation. The time measurement was started right after the hyperspectral image loaded to CPU memory and stopped right after the results were obtained and stored in CPU memory. All the experiments are conducted 10 times and averaged to remove the computer error in Table 1.

**Table 1.** Processing Time (In Seconds) for CPU and GPU Implementation of WSSKRX.

| | **San Diego Airport Dataset** | **Synthetic Dataset** | **SpecTIR Dataset** |
|---|---|---|---|
| CPU/s | 154.097 | 411.388 | 655.076 |
| GPU/s | 11.154 | 16.320 | 20.624 |
| Speedup | 13.815 | 25.208 | 31.763 |

Among the three datasets, the data amount of SpecTIR dataset is the largest and San Diego Airport dataset is the smallest. Plenty of computation in kernel maps and inverse matrices processing make the CPU version of WSSKRX time-consuming. However, it can be observed in Table 1 that a high speedup (up to 31.763) is achieved by the GPUs implementation of which processing time is much lower than the CPU version. The advantage of the implementation on GPUs could be more distinct with the increase of the data size. This is mainly due to the fact that a much larger range of data can be executed effectively.

To sum up, the experiments reported on Table 1 indicate that the GPUs can significantly improve the computational performance of considered target detection algorithms. It provides speedup on the order 13.815 for San Diego Airport dataset, on the order 25.208 for synthetic dataset, and on the order 31.763 for SpecTIR dataset. Although the proposed implementations can still be optimized, significant speedups can be obtained when the algorithms are applied on GPUs.

## 5. Discussion

In the previous section, the simulation and experiment results are analyzed. It is found that the WSSKRX algorithm effectively improves the detection accuracy. This is mainly due to the introduction of the spatial information. Reconstruction of the PUT can balance the relationship between neighborhood pixels via different weights, thereby decrease the influence of background noise. As the growth of the spatial resolution in HSI data, better use of spatial information will be the trend which is the field of anomaly detection.

The calculation principles of GPUs is parallelization of a large number of tasks to achieve and gains ideal parallel efficiency. The results demonstrated that parallel speedup increased as the increase of data size. The recent explosion in the amount and dimensionality of hyperspectral data, calls for the incorporation of parallel computing techniques to accelerate the time-consuming algorithms. Therefore, it is of great academic significance and valuable to do research on GPUs. Generally, the algorithm with spatial has high computational complexity, and by virtue of GPUs, it will obtain a good detection performance.

## 6. Conclusions and Future Research Lines

In this paper, a new weighted spatial-spectral information kernel RX algorithm (WSSKRX) and its parallel implementation on GPUs is presented. The proposed algorithm effectively reduces the interference of the background noise in the detection by utilizing spatial-spectral information rationally, which makes the detection performance better. Taking advantage of the designed parallel systems, the GPU versions of WSSKRX algorithm provide significant speedup when compared with CPU versions. The speedup is more distinct with the increase of the amount of data. Experimental results was oriented towards analyzing the anomaly target detection accuracy and parallel performance with synthetic and real hyperspectral images. In future work we will continue to modify the algorithm to improve its accuracy and explore additional strategies for better computing performance. Other architectures, such as Digital Signal Processors (DSPs) and Field-Programmable Gate Array (FPGA), will be also applied due to their capacity of onboard high performance hyperspectral data processing systems.

**Acknowledgments:** This work is partially supported by National Natural Science Foundation of China under Grant (Grant No. 61405041 and 61571145), the China Postdoctoral Science Foundation (Grant No. 2014M551221), the Heilongjiang Postdoctoral Science Foundation (Grant No. LBH-Z13057), the Key Program of Heilongjiang Natural Science Foundation (Grant No. ZD201216), Program Excellent Academic Leaders of Harbin (Grant No. RC2013XK009003) and the Fundamental Research Funds for the Central Universities (Grant No. HEUCF1708).

**Author Contributions:** All authors participated in the analyzing and writing the paper. Chunhui Zhao conceived and proposed the method. Jiawei Li conducted and analyzed the experiments and wrote the manuscript. Meiling Meng and Xifeng Yao revised the paper in terms of grammar and writing. All authors reviewed and approved the final manuscript.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Bajorski, P. Target detection under misspecified models in hyperspectral images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2012**, *5*, 470–477. [CrossRef]
2. Du, B.; Zhang, L.P. Random-selection-based anomaly detector for hyperspectral imagery. *IEEE Trans. Geosci. Remote Sens.* **2011**, *49*, 1578–1589. [CrossRef]
3. Khazai, S.; Safari, A.; Mojaradi, B.; Homayouni, S. An approach for subpixel anomaly detection in hyperspectral images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2013**, *6*, 769–778. [CrossRef]
4. Ma, L.; Crawford, M.M.; Tian, J. Anomaly detection for hyperspectral images based on robust locally linear embedding. *J. Infrared Millim. Terahertz Waves* **2010**, *31*, 753–762. [CrossRef]
5. Malpica, J.A.; Rejas, J.G.; Alonso, M.C. A projection pursuit algorithm for anomaly detection in hyperspectral imagery. *Pattern Recognit.* **2008**, *41*, 3313–3327. [CrossRef]
6. Yuan, Y.; Wang, Q.; Zhu, G. Fast Hyperspectral Anomaly Detection via High-Order 2-D Crossing Filter. *IEEE Trans. Geosci. Remote Sens.* **2015**, *53*, 620–630. [CrossRef]
7. Yuan, Y.; Ma, D.; Wang, Q. Hyperspectral Anomaly Detection by Graph Pixel Selection. *IEEE Trans. Cybern.* **2016**, *46*, 1–12. [CrossRef] [PubMed]
8. Reed, I.S.; Yu, X. Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution. *IEEE Trans. Signal. Process.* **1990**, *38*, 1760–1770. [CrossRef]
9. Kwon, H.; Nasrabadi, N.M. Kernel RX-algorithm: A nonlinear anomaly detector for hyperspectral imagery. *IEEE Trans. Geosci. Remote Sens.* **2005**, *43*, 388–397. [CrossRef]
10. Dell'Acqua, F.; Gamba, P.; Ferrari, A.; Palmason, J.A. Exploiting spectral and spatial information in hyperspectral urban data with high resolution. *IEEE Geosci. Remote Sens. Lett.* **2004**, *1*, 322–326. [CrossRef]
11. Du, B.; Zhao, R.; Zhang, L.; Zhang, L. A spectral-spatial based local summation anomaly detection method for hyperspectral images. *Signal Process.* **2016**, *124*, 115–131. [CrossRef]
12. Liu, J.; Wu, Z.; Li, J.; Xiao, L. Spatial–Spectral Hyperspectral Image Classification Using Random Multiscale Representation. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2016**, *9*, 1–13. [CrossRef]
13. Ma, X.; Wang, H.; Wang, J. Semisupervised classification for hyperspectral image based on multi-decision labeling and deep feature learning. *ISPRS J. Photogramm. Remote Sens.* **2016**, *120*, 99–107. [CrossRef]
14. Wang, Q.; Lin, J.; Yuan, Y. Salient Band Selection for Hyperspectral Image Classification via Manifold Ranking. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 1–11. [CrossRef] [PubMed]
15. Paz, A.; Plaza, A. Cluster versus GPU implementation of an orthogonal target detection algorithm for remotely sensed hyperspectral images. In Proceedings of the IEEE International Conference on Cluster Computing, Heraklion, Greek, 20–24 September 2010.
16. Wu, X.; Huang, B.; Plaza, A.; Li, Y.; Wu, C. Real-time implementation of the pixel purity index algorithm for endmember identification on GPUs. *IEEE Geosci. Remote Sens. Lett.* **2014**, *11*, 955–959. [CrossRef]
17. Tarabalka, Y.; Haavardsholm, T.V.; Kåsen, I.; Skauli, T. Real-time anomaly detection in hyperspectral images using multivariate normal mixture models and GPU processing. *J. Real Time Image Process.* **2009**, *4*, 287–300. [CrossRef]
18. Quesada-Barriuso, P.; Arguello, F.; Heras, D.B.; Benediktsson, J.A. Wavelet-Based classification of hyperspectral images using extended morphological profiles on graphics processing units. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 2962–2970. [CrossRef]
19. Molero, J.M.; Garzón, E.M.; García, I.; Plaza, A. Analysis and optimizations of global and local versions of the RX algorithm for anomaly detection in hyperspectral data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2013**, *6*, 801–814. [CrossRef]
20. Ruiz, A.; Lopez-De-Teruel, P.E. Nonlinear kernel-based statistical pattern analysis. *IEEE Trans. Neural Netw.* **2001**, *12*, 16–32. [CrossRef] [PubMed]
21. Schölkopf, B.; Smola, A. *Learning with Kernels*; MIT Press: Cambridge, MA, USA, 2002.

22. Camps-Valls, G.; Gomez-Chova, L.; Munoz-Mari, J.; Vila-Frances, J. Composite kernels for hyperspectral classification. *IEEE Geosci. Remote Sens. Lett.* **2006**, *3*, 93–97. [CrossRef]
23. Wu, X.; Guo, B.; Chen, C.; Shen, H. A RX-based hyperspectral target detection method by fusing two kernels. In Proceedings of the 2014 7th International Congress on Image and Signal Processing, Dalian, China, 14–16 October 2014.