

Research Article

Randomized SVD Methods in Hyperspectral Imaging

Jiani Zhang,¹ Jennifer Erway,¹ Xiaofei Hu,¹ Qiang Zhang,² and Robert Plemmons³

¹Department of Mathematics, Wake Forest University, Winston-Salem, NC 27109, USA

²Department of Biostatistical Sciences, Wake Forest School of Medicine, Winston-Salem, NC 27157, USA

³Departments of Mathematics and Computer Science, Wake Forest University, Winston-Salem, NC 27109, USA

Correspondence should be addressed to Robert Plemmons, plemmons@wfu.edu

Received 16 May 2012; Accepted 2 August 2012

Academic Editor: Heesung Kwon

Copyright © 2012 Jiani Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We present a randomized singular value decomposition (rSVD) method for the purposes of lossless compression, reconstruction, classification, and target detection with hyperspectral (HSI) data. Recent work in low-rank matrix approximations obtained from random projections suggests that these approximations are well suited for randomized dimensionality reduction. Approximation errors for the rSVD are evaluated on HSI, and comparisons are made to deterministic techniques and as well as to other randomized low-rank matrix approximation methods involving compressive principal component analysis. Numerical tests on real HSI data suggest that the method is promising and is particularly effective for HSI data interrogation.

1. Introduction

Hyperspectral imagery (HSI) data are measurements of the electromagnetic radiation reflected from an object or scene (i.e., materials in the image) at many narrow wavelength bands. Often, this is represented visually as a cube, where each slice of the cube represents the image at a different wavelength. Spectral information is important in many fields such as environmental remote sensing, monitoring chemical/oil spills, and military target discrimination. For comprehensive discussions, please see, for example, [1–3]. Hyperspectral image data is often represented as a matrix $A \in \mathbb{R}^{m \times n}$, where each entry A_{ij} is the reflection of i th pixel at the j th wavelength. Thus, a column of A contains the entire image at a given wavelength; each row contains the reflection of one pixel at all given wavelengths—often referred to as the *spectral signature* of a pixel.

HSI data can be collected over hundreds of wavelengths—creating truly massive data sets. The transmission, storing, and processing of these large data sets often present significant difficulties in practical situations [1]. Dimensionality reduction methods provide means to deal with the computational difficulties of the hyperspectral data. These methods often use projections to compress a high-dimensional data space represented by a matrix A into a lower-dimensional space represented by a matrix B , which

is then factorized. Such factorizations are referred to as *low-rank* matrix factorizations, resulting in a low-rank matrix approximation to the original HSI data matrix A . See, for example, [2, 4–6].

Dimensionality reduction techniques are generally regarded as lossy compression; that is, the original data is not exactly represented or reconstructed by the lower-dimensional space. For lossless compression of HSI data, there have been efforts to exploit the correlation structure within HSI data plus coding the residuals after stripping off the correlated parts; see, for example, [7, 8]. However, given the large number of pixels, these correlations are often restricted to the spatially or spectrally local areas, while the dimension reduction techniques essentially explore the global correlation structure. By coding the residuals after subtracting the original matrix by its low-dimensional representation, one can compress the original data in a lossless manner, as in [8]. The success of lossless compression requires low entropy of the data distribution, and, as we shall see in the experiments section, generally the entropy of residuals for our method will be much lower than the entropy of the original data.

Low-rank matrix factorizations can be computed using two general types of algorithms: deterministic and probabilistic. The most popular methods for deterministic low-rank factorizations include the singular value decomposition

(SVD) [9] and principal component analysis (PCA) [10]. Advantages of these methods include the following: first, often a small number of singular vectors (or principal components) sufficiently capture the action of a matrix; second, the singular vectors are orthonormal; third, the truncated SVD (TSVD) is the optimal low-rank representation of the original matrix in terms of Frobenius norm by the Eckart-Young theorem [9]. This last advantage is especially suited for compression with the TSVD method, since the Frobenius norm of the residual matrix is the smallest among all rank- k representations of the original matrix, and hence we should expect a much lower entropy in its distributions—making it suitable for compressive coding schemes. Both decompositions offer truncated versions so that these decompositions can be used to represent an n -band hyperspectral image with the data-size-equivalent of only k images, where $k \ll n$. For applications of the SVD and PCA in hyperspectral imaging see, for example, [11, 12].

The traditional deterministic way of computing the SVD of a matrix $A \in \mathbb{R}^{m \times n}$ is typically a two-step procedure. In the first step, the matrix is reduced to a bidiagonal matrix using householder reflections or sometimes combined with a QR decomposition if $m \gg n$. This takes $O(mn^2)$ floating-point operations (flops), assuming that $m \geq n$. The second step is to compute the SVD of the bidiagonal matrix by an iterative method in $O(n)$ iterations, each costing $O(n)$ flops. Thus, the overall cost is still $O(mn^2)$ flops [13, Lecture 31]. In HSI applications, the datasets can easily break into the million-pixel or even giga-pixel level, which renders this operation impossible on typical desktop computers.

One solution is to apply probabilistic methods which give closely approximated singular vectors and singular values, while the complexity is at a much lower level. These methods begin by randomly projecting the original matrix to obtain a lower-dimensional matrix, while the range of the original matrix is asymptotically kept intact. The much-smaller projected matrix is then factorized using a full-matrix decomposition such as SVD or PCA, after which the resulting singular vectors are backprojected to the original space. Compared to deterministic methods, probabilistic methods often offer the lower cost and more robustness in computation, while achieving high-accuracy results. See the seminal paper [14], and the references therein.

Knowing the redundancy of HSI data, especially in the spectral dimension, recently we have observed studies on the compressive HSI sensing, either algorithmic [11, 15] or experimental [6, 16, 17], and all of them involve a random projection of the data onto a lower-dimensional space. For example, in [11] Fowler proposed an approach that exploits the use of compressive projections in sensors that integrate dimensionality reduction and signal acquisition to effectively shift the computational burden of PCA from the encoder platform to the decoder site. This technique, termed compressive-projection PCA (CPPCA), couples random projections at the encoder with a Rayleigh-Ritz process for approximating eigenvectors at the decoder. In its use of random projections, this technique can be considered to possess a certain duality with our approach to randomized SVD methods in HSI. However, CPPCA recovers coefficients of

a known sparsity pattern in an unknown basis. Accordingly, CPPCA requires the additional step of eigenvector recovery.

In this paper, we present a *randomized singular value decomposition* (rSVD) method for the purposes of lossless compression, reconstruction, classification, and target detection. On a large HSI dataset we apply the rSVD method to demonstrate its efficiency and effectiveness of the proposed method. On another HSI dataset, we will show the effectiveness of the proposed algorithm in detecting targets, especially small targets, through singular vectors. In terms of reconstruction quality, we will compare our algorithm with CPPCA [11] by using the signal-to-noise ratio (SNR).

We note that Chen et al. [18] have recently provided an extensive study on the effects of linear projections on the performance of target detection and classification of hyperspectral imagery. In their tests they found that the dimensionality of hyperspectral data can typically be reduced to $1/5 \sim 1/3$ that of the original data without severely affecting performance of commonly used target detection and classification algorithms.

The structure of the remainder of the paper is as follows. In Section 2, we give a detailed overview of rSVD in Section 2.1, the connections between this work and CPPCA in Section 2.2, and the compression and reconstruction of HSI data in Section 2.3. In Section 3, we present numerical results of the rSVD method on two publicly available real data sets. Finally, we draw some conclusions and identify some topics for future work in Section 4.

2. Review of Randomized Singular Value Decomposition

We start by defining terms and notations. The singular value decomposition (SVD) of a matrix $A \in \mathbb{R}^{m \times n}$ is defined as

$$A = U\Sigma V^T, \quad (1)$$

where U and V are orthonormal, and Σ is a rectangular diagonal matrix whose entries on the diagonal are the singular values denoted as σ_i . The column vectors of U and V are left and right singular vectors, respectively, denoted as u_i and v_i . Define the truncated SVD (TSVD) approximation of A as a matrix A_k such that

$$A_k = \sum_{i=1}^k \sigma_i u_i v_i^T. \quad (2)$$

We define the randomized SVD (rSVD) of A as follows:

$$\hat{A}_k = \hat{U} \hat{\Sigma} \hat{V}^T, \quad (3)$$

where \hat{U} and \hat{V} are both orthonormal, and $\hat{\Sigma}$ is diagonal with diagonal entries denoted as $\hat{\sigma}_i$. Denote the column vectors of \hat{U} and \hat{V} as \hat{u}_i and \hat{v}_i , respectively, and call them randomized singular vectors. Here, u_i , v_i , and σ_i are related to \hat{u}_i , \hat{v}_i , and $\hat{\sigma}_i$, respectively. Define the residual matrix of a TSVD approximation as follows:

$$R_k = A - A_k, \quad (4)$$

```

Input: An  $m \times n$  matrix  $A$  and a precision measure  $\epsilon$ .
Output: An  $m \times k$  matrix  $Q$  and rank  $k$ .
Initialize  $Q$  as an empty matrix,  $e = 1$  and  $k = 0$ .
while  $e > \epsilon$  do
  (1)  $k = k + 1$ .
  (2)  $y_i = A\omega_i$ , where  $\omega_i$  is a Gaussian random vector.
  (3)  $q_i = (I - QQ^T)y_i$ .
  (4)  $q_i = q_i/\|q_i\|$ .
  (5)  $Q \leftarrow [Q \ q_i]$ .
  (6)  $\Omega \leftarrow [\Omega \ \omega_i]$ .
  (7) Compute error  $e = \|A - QQ^T A\|_F/\|A\|_F$ .
end

```

ALGORITHM 1: Construct an orthonormal matrix Q that approximates the range of matrix $A\Omega$.

and the residual matrix of a rSVD approximation as follows:

$$\hat{R}_k = A - \hat{A}_k. \quad (5)$$

Define the random projection of a matrix as follows:

$$Y = \Omega^T A, \quad \text{or} \quad Y = A\Omega, \quad (6)$$

where Ω is a random matrix with independent and identically distributed (i.i.d.) entries.

2.1. Randomized SVD Algorithm. The rSVD algorithm as considered by [14] explores approximate matrix factorizations using random projections, separating the process into two stages. In the first stage, random sampling is used to obtain a reduced matrix whose range approximates the range of A ; in the second stage, the reduced matrix is factorized. In this paper, we use this framework for computing the rSVD of a matrix A .

The first stage of the method is common to many approximate matrix factorization methods. For a given $\epsilon > 0$, we wish to find a matrix Q with orthonormal columns such that

$$\|A - QQ^T A\|_F^2 \leq \epsilon. \quad (7)$$

Algorithm 1 [14] can be used to compute Q .

Because in practice we may not know the target rank k of A , Algorithm 1 allows us to look for an appropriate target rank based upon given ϵ such that (7) holds. However if the target rank is known, one can avoid computing the error term e at each iteration by replacing the *While* loop with a *For* loop. In practice, the number of columns of Q is usually chosen to be slightly larger than the numerical rank of A [14]. Without loss of generality, we assume that $Q \in \mathbb{R}^{m \times l}$, where $l \ll n$. The columns of Q form an orthogonal basis for the range of $A\Omega$, where Ω is a matrix composed of the random vectors $\{\omega_i\}$, typically with a standard normal distribution [14]. The range of the product $A\Omega$ is an approximation to the range of A .

The second stage of the rSVD method is to compute the SVD of the reduced matrix $Q^T A \in \mathbb{R}^{l \times m}$. Since $l \ll n$, it is generally computationally feasible to compute the SVD of

the reduced matrix. Letting $\tilde{U}\hat{\Sigma}\hat{V}^T$ denote the SVD of $Q^T A$, we obtain that

$$A \approx (Q\tilde{U})\hat{\Sigma}\hat{V}^T = \hat{U}\hat{\Sigma}\hat{V}^T, \quad (8)$$

where $\hat{U} = Q\tilde{U}$ and \hat{V} are orthogonal matrices, and thus by (8), $\hat{U}\hat{\Sigma}\hat{V}^T$ is an approximate SVD of A , and the range of \hat{U} is an approximation to range of A . Algorithm 2 summarizes the discussion above. See [14] for details on the choice of l , along with extensive numerical experiments using randomized SVD methods and a detailed error analysis of the two-stage method described above.

Next we discuss several variations of Algorithm 2 depending on the properties of A . We will test all cases in the numerical results section.

Case 1. If knowing the target rank k , and if the singular values of A decay rapidly, we can skip Algorithm 1 by simply using the rank revealing QR factorization, $Y = QR$, where Q is an orthogonal basis of the range of Y . Figure 1 from [19] compares the approximation error e_k and the theoretical error σ_{k+1} of a matrix A , and clearly when the singular values of A decay rapidly, e_k is close to the theoretical error σ_{k+1} with high probability.

Case 2. If the singular values of A decay gradually, or σ_k/σ_1 is not small, we may lose the accuracy of estimates. Consider introducing a power q and forming Y as $Y = (AA^T)^q A$. Since $(AA^T)^q A$ has the same singular vectors as A , while its singular values, $\{\sigma_i^{2q+1}, i = 1, \dots, n\}$, decay more rapidly. Hence the error will be smaller by Theorems 2.3 and 2.5 in [14]. From Figure 2, we see that the e_k is not always close to σ_{k+1} , especially when $q = 0$, but, by increasing the power q , we observe the reduction of errors.

Case 3. Algorithm 2 requires us to revisit the input matrix, while this may be not feasible for large matrices. For example, in ultraspectral imaging [20], one could have thousands of spectral bands, and PCA on such datasets would require computing the eigenvectors and eigenvalues of a covariance matrix with a huge dimension. Another example is in the atmospheric correction model called MODTRAN5 [21], that utilizes large lookup tables (LUTs), and the compression

Input: An $m \times n$ matrix A and rank k with $k \leq n \leq m$.
Output: The rSVD of A : $\hat{U}, \hat{\Sigma}, \hat{V}$.
(1) Generate a Gaussian random matrix $\Omega \in \mathbb{R}^{n \times k}$.
(2) Form the projection of A : $Y = A\Omega$.
(3) Construct $Q \in \mathbb{R}^{m \times k}$ by Algorithm 1.
(4) Set $B = Q^T A \in \mathbb{R}^{k \times n}$.
(5) Compute the SVD of B , $B = \tilde{U} \hat{\Sigma} \hat{V}^T$.
(6) $\hat{U} = Q\tilde{U}$.

ALGORITHM 2: The basic rSVD algorithm.

Input: An $m \times n$ matrix A and an integer J .
Output: $\{B_j, Q_j, \hat{r}_j, j = 1, \dots, J\}$.
 $j = 1$
while *flight continues* **do**
(1) Acquire the HSI data, A_j , scanned in the last few seconds.
(2) Apply Algorithm 1 for Q_j and B_j .
(3) Compute the residual, $r_j = A_j - Q_j B_j$.
(4) Code r_j as \hat{r}_j with a parallel floating point coding algorithm.
(5) Store Q_j, B_j and compressed r_j .
(6) $j = j + 1$.
end

ALGORITHM 3: rSVD encoder.

of LUTs by the PCA technique would again require the eigen decomposition of large covariance matrices. Here we introduce a variation of Algorithm 2 that only requires one pass over a large symmetric matrix. Now we define matrix B as follows:

$$B = Q^T A Q, \quad (9)$$

and we multiply by $Q^T \Omega$, that is,

$$B Q^T \Omega = Q^T A Q Q^T \Omega. \quad (10)$$

Since $A \approx A Q Q^T$, we have the following approximation:

$$B Q^T \Omega \approx Q^T A \Omega = Q^T Y, \quad (11)$$

and hence by a least-square solution we have

$$B \approx Q^T Y (Q^T \Omega)^\dagger, \quad (12)$$

where the superscript \dagger represents the pseudoinverse. Notice the absence of A in the approximate formula of B . Thus, for a large symmetric A , we will use (12) rather than $Q^T A$ to compute B , while the rest of Algorithm 2 would remain the same.

2.2. Connections to CPPCA. A significant difference between the compressive-projection PCA (CPPCA) approach and our work is that CPPCA uses a random orthonormal matrix P to compress the data matrix A . In comparison, though we also use random projections, the orthonormal matrix Q is constructed from, and directly related to, the data

matrix A . In particular, we compute an orthogonal Q such that $\|A - Q Q^T A\|_F \leq \epsilon$. Also, because the projection onto convex sets (POCSs) algorithm is used for reconstruction, the projection matrix P of CPPCA has to be different for different blocks of the scene, which have to be independently drawn and orthogonalized; meanwhile, one random projection matrix Ω in rSVD is sufficient and can be applied to the whole dataset. Another restriction of CPPCA lies in the fact that the Rayleigh-Ritz method requires well-separated eigenvalues [22], which might be true for the first few largest eigenvalues, but usually not true for the smaller eigenvalues. In a later section we present our approach for matrices with slowly decaying singular values in Case 2.

2.3. Compression and Reconstruction of HSI Data by rSVD.

The flight times of airplanes carrying hyperspectral scanning imagers are usually limited by the data capacity, since within 5 to 10 seconds hundreds of thousands of pixels of hyperspectral data are collected [1]. Hence for real-time onboard processing, it would be desirable to design algorithms capable for processing this amount of data within 5 to 10 seconds before the next section of the scene is scanned. Here we use the proposed rSVD algorithm to losslessly compress blocks of HSI data, each within a frame of 10 second flight time, which is equivalent to dividing the HSI data cube along the flight direction, either the x or y direction, with the number of rows (y direction) or columns (x direction) determined by the ground sample distance (GSD) and the flight speed. Algorithm 3 describes the rSVD encoder, which outputs $\{B_j, Q_j, \hat{r}_j, j = 1, \dots, J\}$ to be stored onboard, where B_j and Q_j are the outputs of Algorithm 2 for

```

Input:  $\{B_j, Q_j, \hat{r}_j, j = 1, \dots, J\}$ .
Output: Reconstructed matrix  $\hat{A}$ .
For  $j = 1 : J$  do
  (1) Decode  $r_j$  from  $\hat{r}_j$  with a parallel floating point decoding algorithm.
  (2)  $A_j = Q_j B_j + r_j$ .
  (3)  $j = j + 1$ .
end
Group all  $\{A_j, j = 1, \dots, J\}$  together in  $A$ .

```

ALGORITHM 4: rSVD decoder.

the j th block of data, while \hat{r}_j is the coded residual. These are then used by Algorithm 4 to reconstruct the original data losslessly, and we can see it only involves a one-pass matrix-matrix multiplication and is without iterative algorithms. Compared to CPPCA, the number of bytes used for storing the B s and Q s is smaller, and the reconstruction only involves matrix-matrix multiplication. The only possible bottleneck might be the residual coding, but the recent development in floating point coding has seen throughputs reaching as much as 75 Gb/s [23] on a graphic processing unit (GPU), while even on an eight-Xeon-core computer we have seen throughput at 20 Gb/s, and both would be sufficiently fast to code the required amount of HSI data within 10 seconds.

3. Numerical Experiments

3.1. Accuracy of the rSVD Estimates. In this section, we will first compare the results from rSVD and from the exact TSVD by the MATLAB function, “*svds*,” which computes the largest k singular values and the associated singular vectors of a large matrix. It is considered to be an efficient and accurate method to obtain the TSVD. To simulate large HSI datasets, we generate random test matrices $A \in \mathbb{R}^{m \times n}$, with n fixed at 100 representing 100 spectral channels, while $m = 100,000; 200,000; \dots, 2,000,000$, representing the number of pixels. The singular values of A are simulated as following a power decay with the power set as -1 , that is, $\sigma_k/\sigma_1 = 1/k$. We will use Algorithm 2 to compute the rSVD. The comparison of computation time is shown in Figure 3(a), from which we find that “*svds*” is almost as effective as rSVD when n is relatively small. However, when m increases, the computation times of “*svds*” increase at a much faster pace than that of rSVD, and note that when $m = 300,000$, the processing time of rSVD is well within 10 seconds, meeting the onboard processing time limit. To judge the accuracy of estimated singular vectors, we compute the correlation or the inner product of singular vectors in U by “*svds*” and \hat{U} by rSVD as shown in Figure 3(b), where we clearly see that both sets are almost identical up to the fifteenth singular vector. To judge the accuracy of estimated singular values, we compute the relative absolute errors, $|\hat{\sigma}_k - \sigma_k|/|\sigma_k|$, and plot them in Figure 3(c). Again we observe the high accuracy of the estimates up to the first 15 singular values. In most HSI datasets, the singular values decay rate is generally faster than $1/k$, and hence we should expect even higher accuracy of the estimates. Also, it is sufficient to estimate the first 15 or even 10

singular vectors and singular values, which would often cover more than 90% of the original variance of the HSI data [1].

Then we numerically test the three special cases discussed in Section 2.1.

Case 1. We have considered square Toeplitz matrices with increasingly large sizes, $n = 15, 30, \dots, 1500$. Figure 4 shows the rapid decay of a $1,000 \times 1,000$ matrix, and hence they are suitable for testing the algorithm. Figure 5(a) shows that the relative Frobenius norm errors rise and fall in the order of 10^{-12} and remain in the same order even when the size of a matrix increases. Figure 5(b) demonstrates that the computational time of rSVD is very short for the Toeplitz matrices whose singular values decay rapidly.

Case 2. Here we simulate $10,000 \times 100$ matrices with slowly decaying singular values, that is, $\sigma_i/\sigma_1 = i^{-s}$, with $s = 0.2, 0.4, \dots, 1.0$. For each matrix, we run the rSVD algorithm 100 times for each power q in the set, $\{q = 1, 2, \dots, 20\}$. The norm errors of reconstructed matrices are averaged across 100 runs and normalized by the norm error when $q = 1$, that is, e_q/e_1 . Figure 6 shows that increasing the power q improves the reconstruction quality or decreases the norm error, and greater effects are observed for larger s because the singular values of $(AA^T)^q$ are $\sigma_i^{2q} = \sigma_1^{2q} i^{-2qs}$.

Case 3. To simulate covariance matrices, we generate a sequence of positive definite symmetric matrices with increasing size as $n = 100, 110, \dots, 2,000$. The eigen-spectrum follows a power decay with the power set as -1 . We apply the modified B as in Case 3 to compute its SVD, rather than using $Q^T A$. We set $k = 25$. Figure 7(a) shows the computation time compared with using regular SVD in MATLAB, while Figure 7(b) shows the relative Frobenius norm errors between the original matrix and its low-rank approximation. Apparently the computation time used by rSVD is far less than the regular SVD, while the accuracies are quite high.

3.2. rSVD on a Large HSI Dataset and a Lossless Compression. The rSVD algorithm was also applied to a relatively large HSI dataset consisting of a $920 \times 4,933 \times 58$ image cube collected over Gulfport MS by a commercial hyperspectral sensor having a spectral range of 0.45 to 0.72 microns. This cube was then unfolded into a large matrix of size $4,538,360 \times 58$. Running an exact SVD algorithm is almost impossible on a regular desktop computer with limited memory and speed,

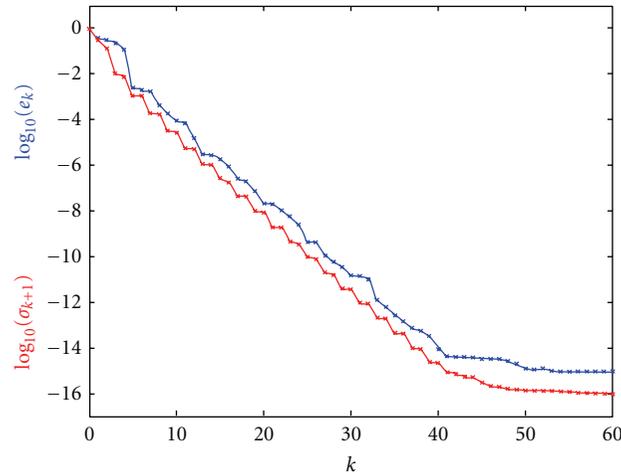


FIGURE 1: Comparison of the computed error e_k (blue) and the theoretical error σ_{k+1} (red).

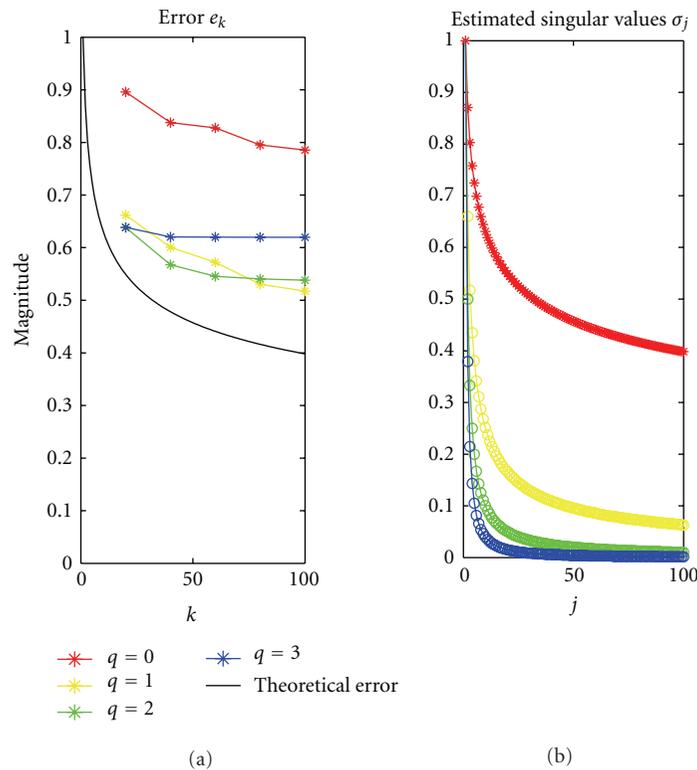


FIGURE 2: Comparison of the error e_k and the theoretical error σ_{k+1} . The red curve shows that the error e_k is greater than the theoretical error σ_{k+1} . Note that the singular values decay more rapidly as q increases.

while the rSVD algorithm gives us singular values and vectors very close to the true ones, with a significantly reduced amount of flops and memory required. For the first 25 singular vectors and singular values, it only takes 68 seconds on a desktop computer with Xeon 3.2 GHz Quadcores and 12 GB memory. From the computed singular values and vectors, we observed that the singular vectors after the ninth singular vector all appear to be noise, indicating that the data matrix does have a low-rank representation.

Figure 8 shows the results for a small scene from the large dataset described above, consisting of part of the University of Southern Mississippi Campus, extracted from the large Gulfport MS dataset. Notice the targets placed in the scene, for detection and identification tests. The first eight singular vectors, \hat{u}_i , are folded back from the transformed data. The first singular vector shown in Figure 8 is the mean image across 58 spectral bands, while the second singular vector shows high intensity at the grass and foliage pixels, the third

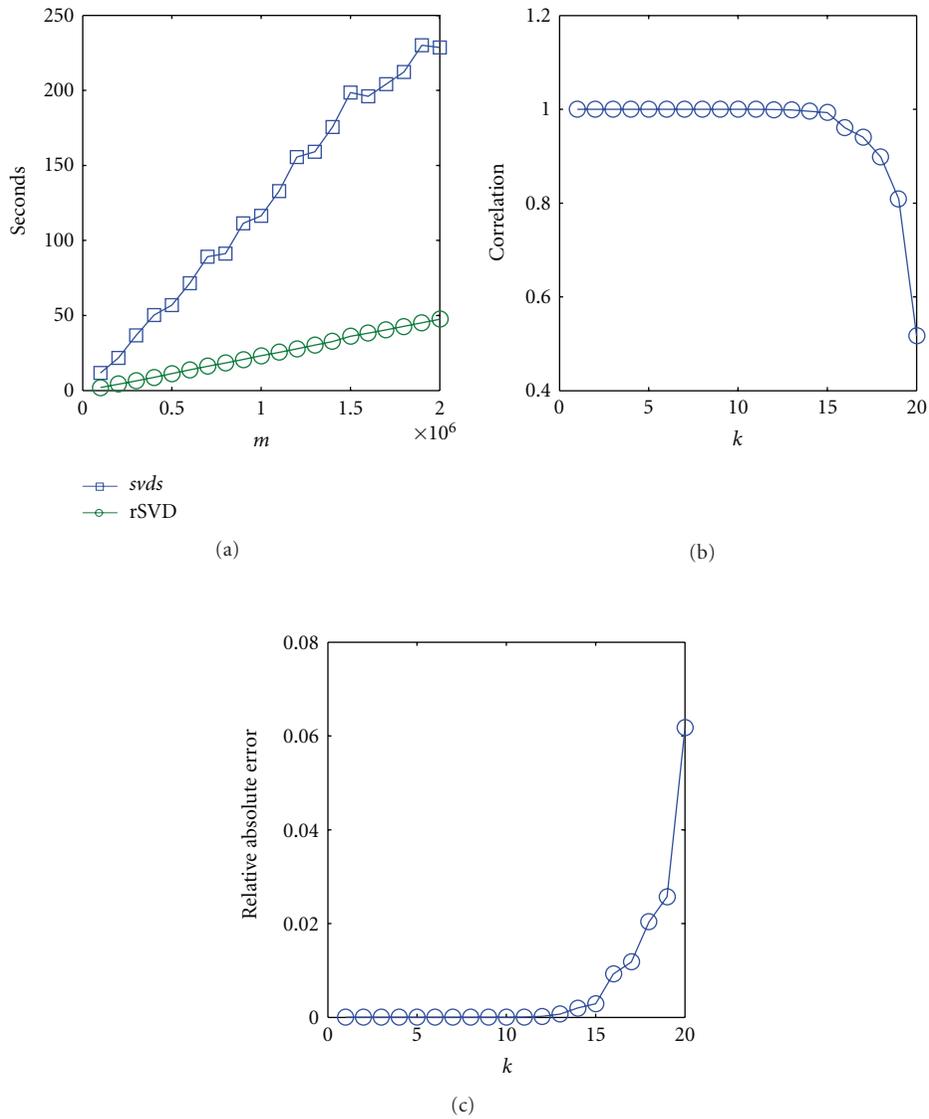


FIGURE 3: (a) Computation time of “svds” and rSVD. (b) Correlations between the singular vectors in U by “svds” and rSVD. (c) Relative absolute differences between the singular values estimated by “svds” and rSVD.

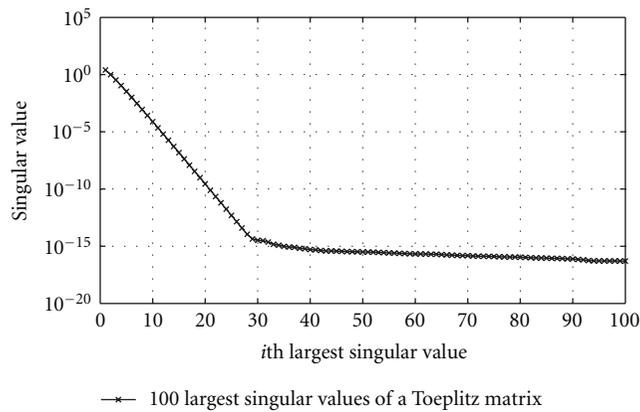


FIGURE 4: Illustration rapidly decaying singular values of matrix of size 1000×1000 .

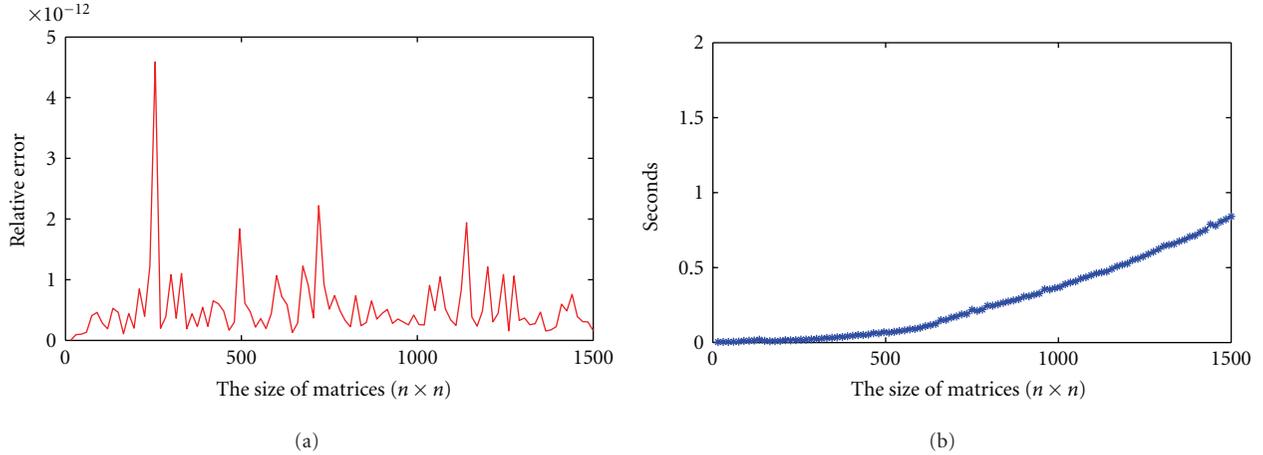


FIGURE 5: (a) The relative errors between the rSVD low-rank approximation and the original matrix A are shown by the red curve. (b) The computational time for rSVD.

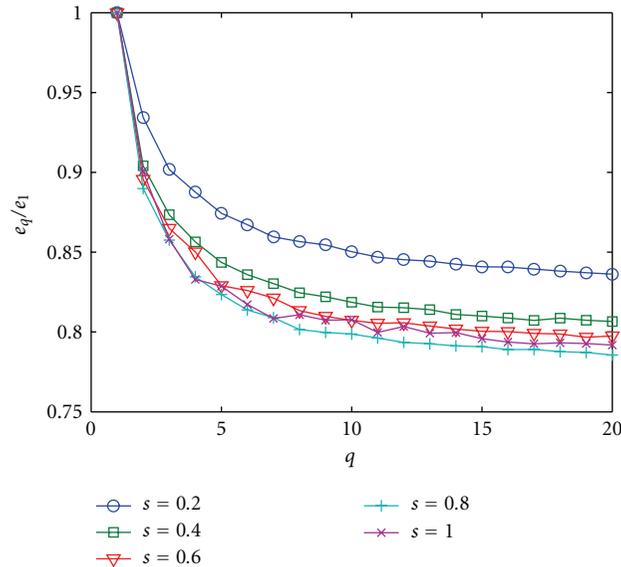


FIGURE 6: The reduction of norm error by increasing the power q for singular value spectrums decaying with various powers s .

shows the targets quite clearly, as well as high-reflectance sandy areas or rooftops, the fifth shows the low-reflectance pavement or roof tops, and shadows, and the seventh shows vehicles at various places marked by the circles. Starting from the eighth, the rest of the singular vectors appear to be mostly noise.

In Figure 9(b), the histogram of entries in \hat{R}_k shows that residuals are roughly distributed as a Laplacian distribution, and all residuals are within the range of $[-.1, .1]$, which is significantly smaller than the original range of A in Figure 9(a). Moreover, most of the residuals (93%) are within the range $[-.01, .01]$ (notice the log scale on the y axis), which means that the entropy of residuals is significantly smaller than the entropy of the original. This justifies a further coding step on the residuals so as to complete a lossless compression scheme. Here we apply the Hoffman

coding due to its fast computation and show the compression ratios at various error rates, corresponding to the numbers of bits required to code the residuals. For example, a 16-bit coding would result in an error in the range of 10^{-5} . Figure 10 provides us options on balancing compression with accuracy. For practical purposes, an error rate in the order of 0.001 might be sufficient, and this would result in a compression ratio of 2.5 to 4. For comparison purpose, the 3D-SPECK [7] on a small dataset of size $320 \times 360 \times 58$ results in a compression ratio of 1.12 at the 16-bit coding. If more sophisticated coding algorithms than Hoffman coding are applied here, we could see more improvements on the compression ratios. For computing the compression ratios, we have assumed 16-bit coding (2-byte) for all the matrices, including B_j , Q_j , the residual matrix and the coded (compressed) residual matrix.

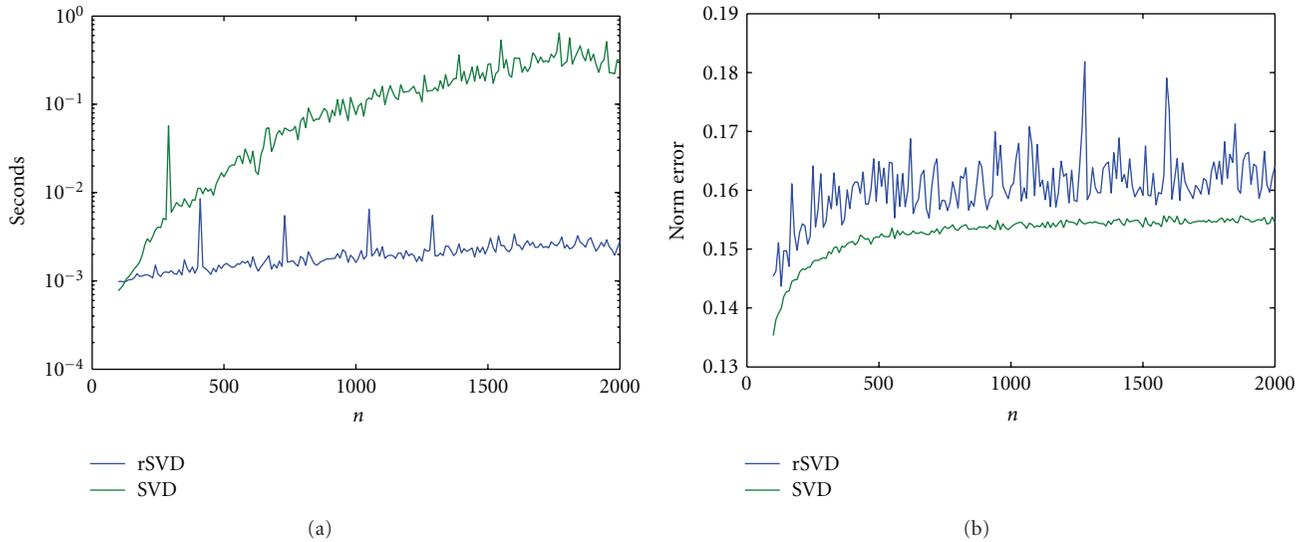


FIGURE 7: (a) Computation time of rSVD in blue and SVD in green. (b) Relative norm errors by rSVD and SVD.

To test the suitability of the onboard real-time processing by Algorithm 3, we apply the rSVD on a $300,000 \times 100$ matrix and see it is finished in 7 seconds on a low-end dual-core laptop computer, and if, with a parallel coding algorithm for the residuals, we should finish Algorithm 3 within the required 10-seconds time frame.

3.3. Small Target Detection Using rSVD. For a small target detection experiment using rSVD, we choose a version of the Forest Radiance HSI dataset, which has been analyzed by using numerous target detection methods, see, for example, [18, 24–26]. Our rationale behind using an SVD algorithm in target detection lies in the fact that even though targets might be of small size, if all the spectrally similar targets have sufficient presence, some singular vectors of the HSI data matrix will reflect these features, and hence the presence of targets can be simply shown by these singular vectors. After removing the water-absorption and other noisy bands, we unfold the $200 \times 150 \times 169$ data cube into a $30,000 \times 169$ matrix and apply Algorithm 2 for the singular vectors \hat{u}_i . Figure 11 shows the sum of first twelve \hat{u}_i folded back into a 200×150 matrix, and we can clearly detect 25 of the 27 targets, while the other two are slightly visible.

3.4. Comparison between rSVD and CPPCA. In this section, we will compare rSVD with CPPCA from the aspects of accuracy and computation time, first on simulated data and then on a real HSI dataset. We first simulate a set of matrices with increasing number of rows (pixels), $m = 10,000, 20,000, \dots, 100,000$, while fixing $n = 100$ as 100 spectral bands. The singular value spectrum is simulated as following a power decay rate with the power set as -1 . Both CPPCA and rSVD algorithms are applied to each simulated matrix, and results are compared in terms of their reconstruction quality and the computation time. Figure 12(a) shows that the running time of rSVD increases linearly with m , while that of CPPCA remains constant,

TABLE 1: Computation times (seconds) of rSVD and CPPCA.

k/n	0.1	0.15	0.2	0.3	0.4	0.5
rSVD	0.212	0.292	0.390	0.707	0.897	1.264
CPPCA	0.247	0.305	0.331	0.368	0.399	0.509

which is not surprising since CPPCA mainly works on eigenvectors of fixed dimension n . However in terms of reconstruction quality, Figure 12(b) shows the advantage of rSVD. Here we set the number of reconstructed eigenvectors by CPPCA to 3 since it provides the best norm errors, while for rSVD we set it to 25.

For the real dataset, we use a small section of the Gulfport dataset and fold it into a matrix A with size 115200×58 . From the reconstructed matrices \hat{A} by both methods, with varying rank k we compare their reconstruction qualities in terms of signal-to-noise ratio (SNR) in Figure 13, and the computation time in Table 1. Again we observe the better reconstruction quality though slightly slower computation of rSVD when compared to CPPCA.

Next, we compare the accuracy of the reconstruction of the eigenvectors, \hat{v}_i , of the covariance matrix of A by these two methods. Given that PCA is an extremely useful tool in HSI data analysis, for example, for classification and target detection, it is essential to obtain a quality reconstruction of the eigenvectors \hat{v}_i by rSVD in terms of accuracy and efficiency. Here we simulate a $10,000 \times 100$ matrix A with orthogonalized random matrices, U_o and V_o , and a power-decay singular value spectrum in Σ_o , with the power set as -1 . Then we run both algorithms for 1,000 times, and, within each time, we compute the angles between eigenvectors by CPPCA and the true ones, and between eigenvectors by rSVD and the true ones. The first row of Figure 14 shows the histograms of angles between the first eight reconstructed eigenvectors by CPPCA and the true ones, while the second row shows the histograms

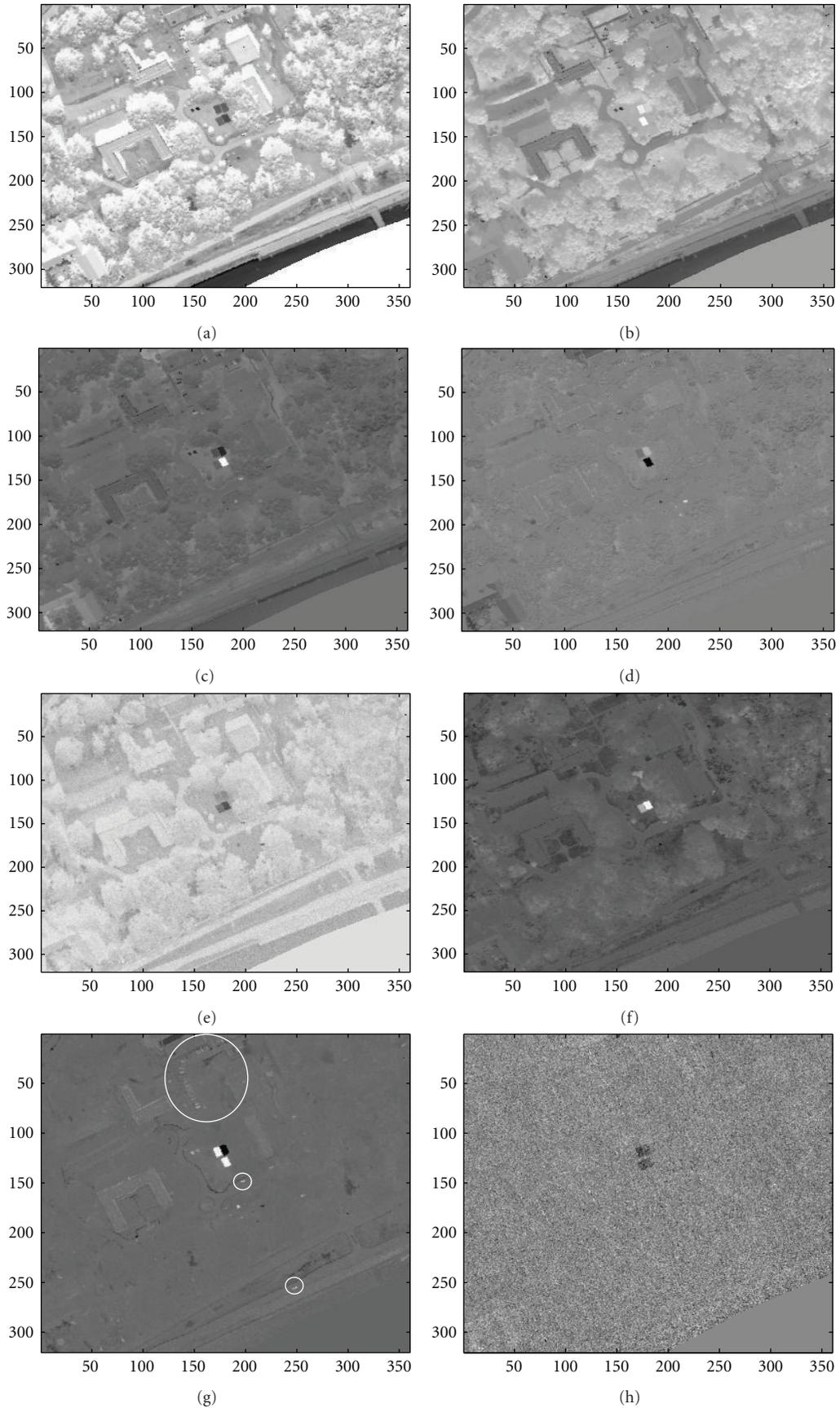


FIGURE 8: The first eight singular vectors, \hat{u}_i , are turned into images. The circles on the image of the seventh singular vectors indicate identified vehicles.

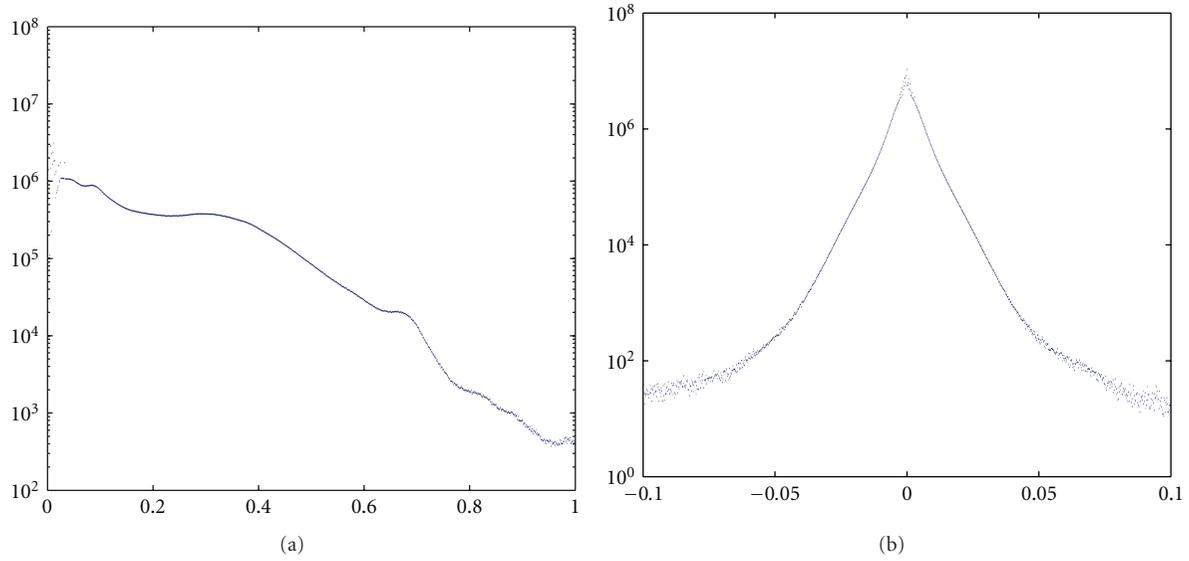


FIGURE 9: (a) The distribution of intensities of the original Gulfport HSI cube. (b) The distributions of residuals after subtracting the TSVD from the original.

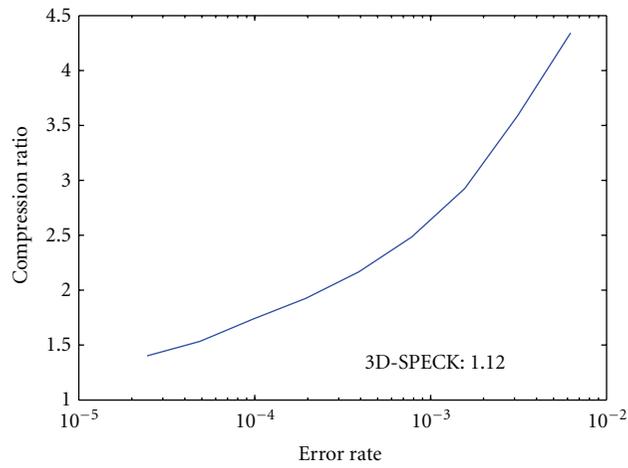


FIGURE 10: The compression ratio as a function of error rate.

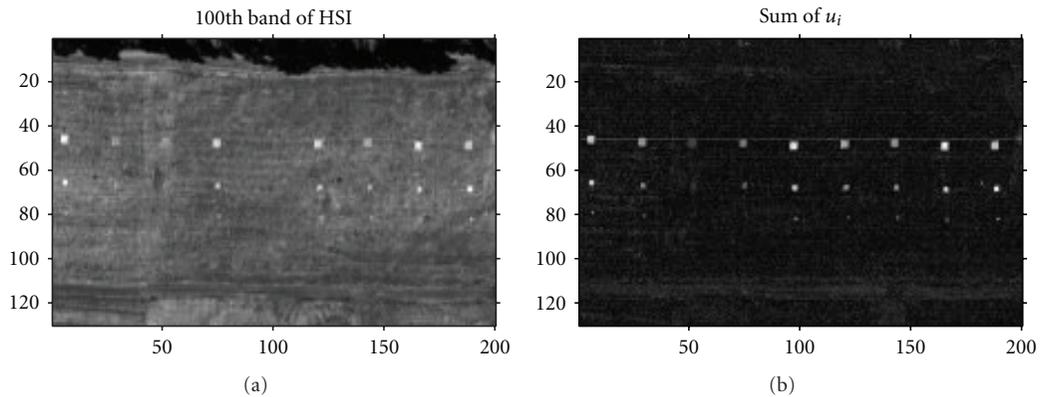


FIGURE 11: One band of the Forest Radiance HSI dataset is shown on the left. Binary target detections are shown on the right, obtained after a summation of the first 12 \hat{u}_i .

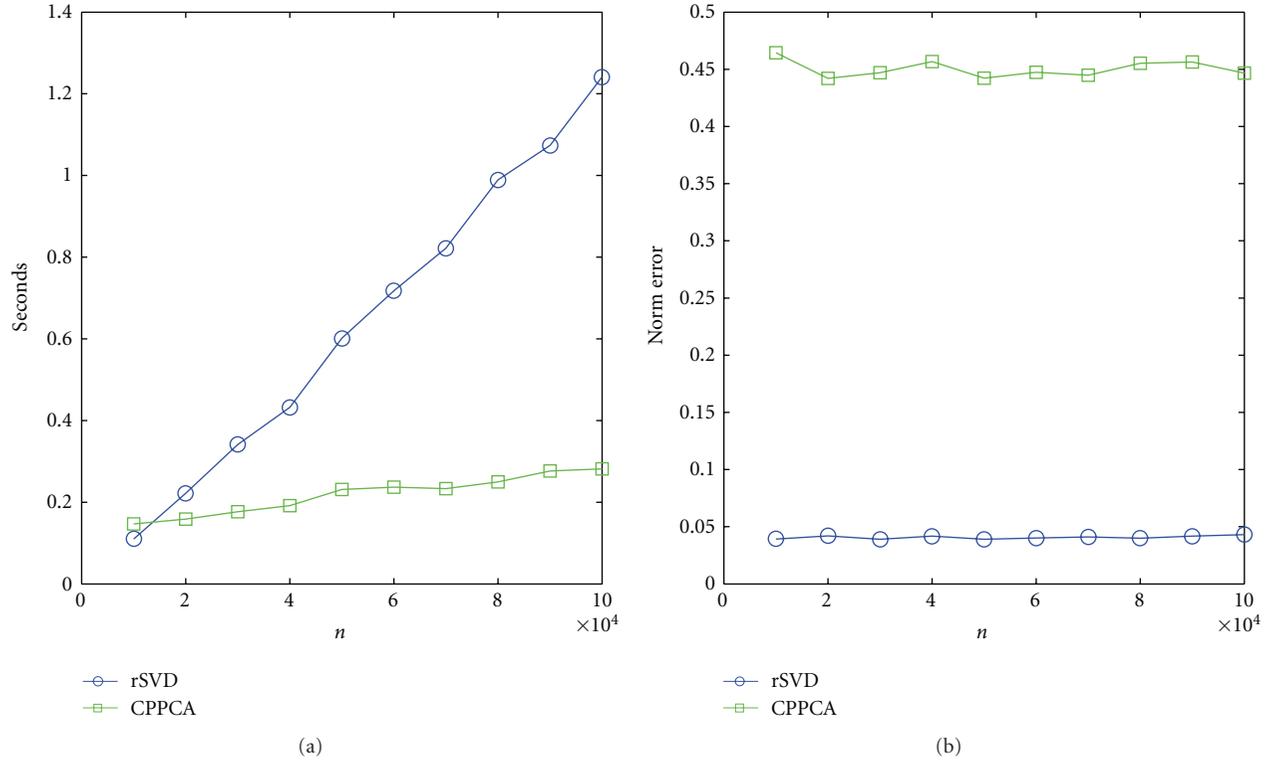


FIGURE 12: (a) Running times of rSVD and CPPCA. (b) Reconstruction qualities of rSVD and CPPCA.

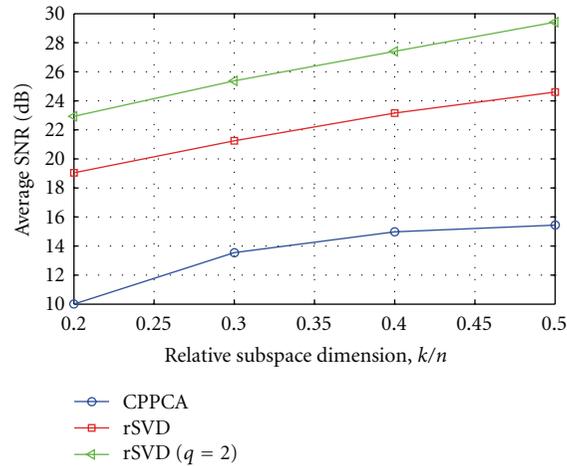


FIGURE 13: Comparison of reconstruction qualities of rSVD and CPPCA in terms of SNR.

of the angles between the first eight eigenvectors by rSVD and the true ones. We can see that the first three or four eigenvectors by CPPCA appear to be close to the true ones, while the rest are not. Hence if using more than four eigenvectors reconstructed by CPPCA, we observe a decrease in reconstruction quality or an increase in the norm error. However in the second row, we see good accuracy of the eigenvectors computed by rSVD.

3.5. *Classification of HSI Data by rSVD.* Since the projection of a HSI data matrix by its truncated singular matrix, that is,

$$A_P = AV_k, \quad (13)$$

contains most of the information in the original matrix A , we can use any classification algorithm, such as the popular k means, to classify HSI data, but also use its representation

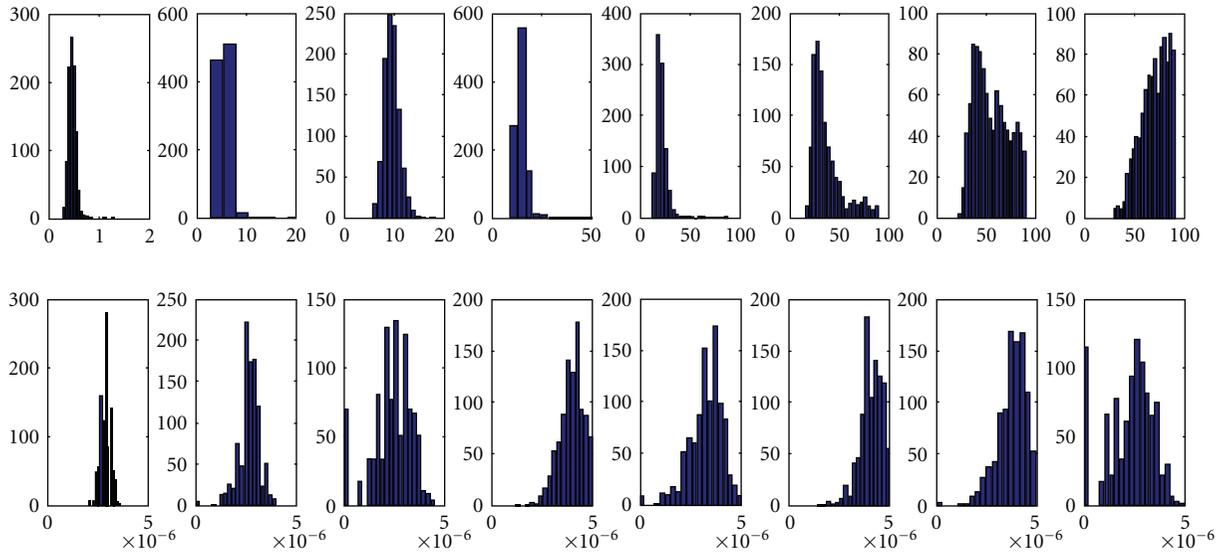
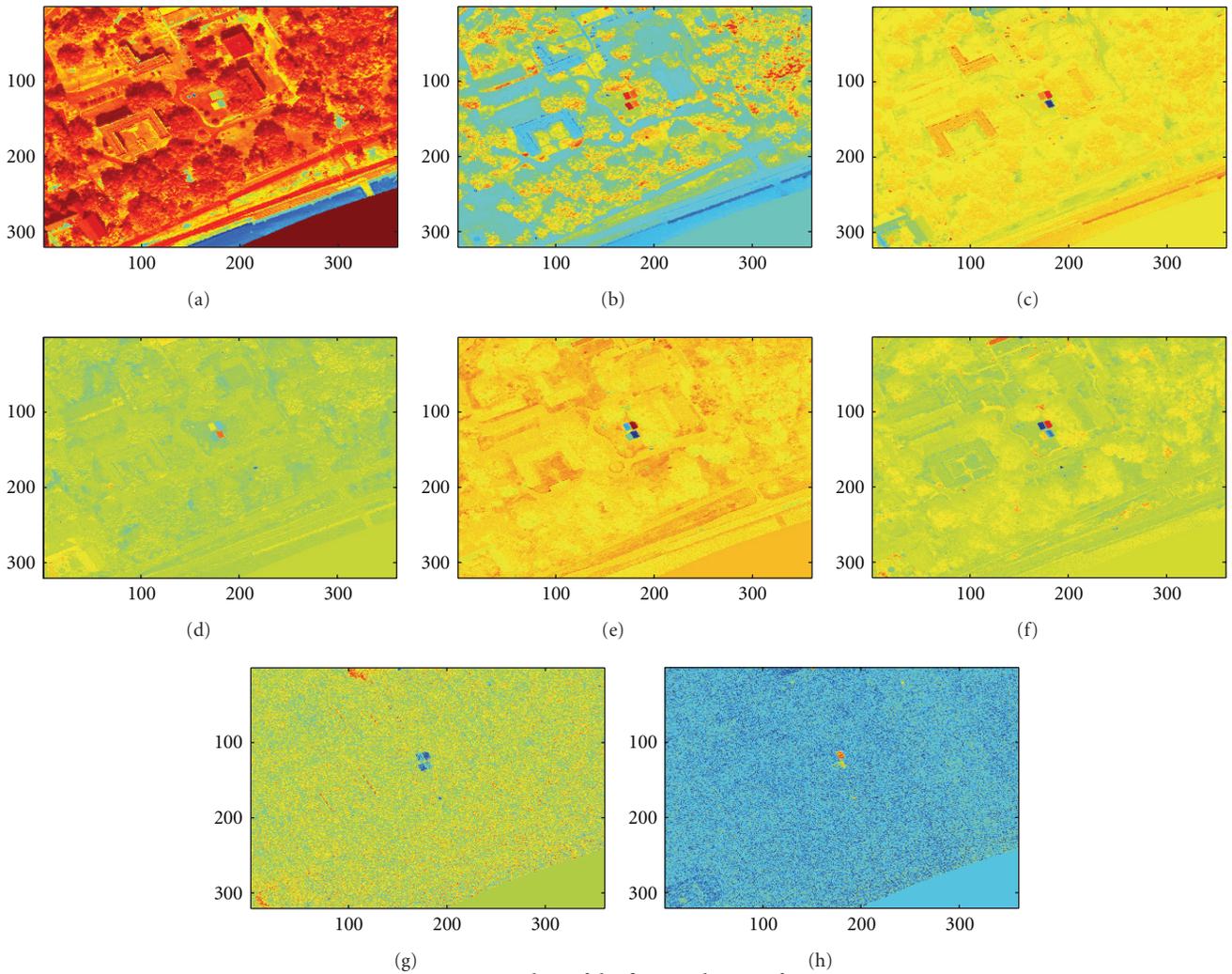


FIGURE 14: Comparison of reconstructed eigenvectors by rSVD and CPPCA with the true ones.



(g) (h) FIGURE 15: Plots of the first 8 columns of A_p .

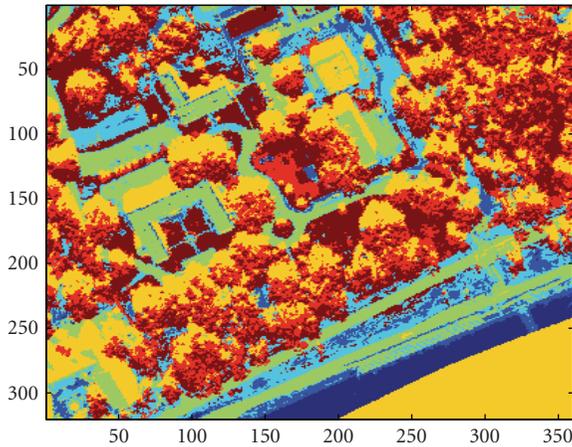


FIGURE 16: The classification result of k -means using A_k .

in a lower-dimensional space. Consider a small section of the Gulfport dataset. Figure 15 shows the first 8 columns of A_P . From the first subfigure, we see that most information of the hyperspectral image is contained in the first column, while the second column almost contains the rest of the information which the first column does not contain. The rest of the columns contain information at more detailed and spatially clustered levels. Figure 16 shows the result of classification by k -means, where we can see the low-reflectance water and shadows in yellow, the foliage in red, the grass in dark red, the pavement in green, high-reflectance beach sand in dark blue, and dirt/sandy grass in blue and light blue.

A comparison with results from running k -means on the full dataset shows that only 13 pixels of all the $320 \times 360 = 115,200$ pixels are classified differently between the full dataset and its low-dimensional representation. Hence it is highly suitable to use this low-dimensional representation for classification.

4. Conclusions

As HSI data sets are growing increasingly massive, compression and dimensionality reduction for analytical purposes has become more and more critical. The randomized SVD algorithms proposed in this paper enable us to compress, reconstruct, and classify massive HSI datasets in an efficient way while maintaining high accuracy in comparison to exact SVD methods. The rSVD algorithm is also found to be effective in detecting small targets down to single pixels. We have also demonstrated the fast computation in compression and reconstruction of the proposed algorithms on a large HSI dataset in an urban setting. Overall, the rSVD provides a lower approximation error than some other recent methods and is particularly well suited for compression, reconstruction, classification, and target detection.

Acknowledgments

The authors wish to thank the referees and the project sponsors for providing very helpful comments and suggestions for

improving the paper. Research by Jiani Zhang and Jennifer Erway was supported by NSF grant DMS-08-11106. Research by Robert Plemmons and Qiang Zhang was supported by the U.S. Air Force Office of Scientific Research (AFOSR), award number FA9550-11-1-0194, and by the U.S. National Geospatial-Intelligence Agency under Contract HM1582-10-C-0011, public release number PA Case 12-433.

References

- [1] M. T. Eismann, *Hyperspectral Remote Sensing*, SPIE Press, 2012.
- [2] H. F. Grahm and E. Paul Geladi, *Techniques and Applications of Hyperspectral Image Analysis*, Wiley, 2007.
- [3] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon et al., "Hyperspectral unmixing overview: geometrical, statistical, and sparse regression-based approaches," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 2, pp. 354–379, 2012.
- [4] J. C. Harsanyi and C. I. Chang, "Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 32, no. 4, pp. 779–785, 1994.
- [5] A. Castrodad, Z. Xing, J. Greer, E. Bosch, L. Carin, and G. Sapiro, "Learning discriminative sparse models for source separation and mapping of hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, pp. 4263–4281, 2011.
- [6] C. Li, T. Sun, K. F. Kelly, and Y. Zhang, "A compressive sensing and unmixing scheme for hyperspectral data processing," *IEEE Transactions on Image Processing*, vol. 21, no. 3, pp. 1200–1210, 2012.
- [7] X. Tang and W. Pearlman, "Three-dimensional wavelet-based compression of hyperspectral images," *Hyperspectral Data Compression*, pp. 273–308, 2006.
- [8] H. Wang, S. D. Babacan, and K. Sayood, "Lossless hyperspectral-image compression using context-based conditional average," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 12, pp. 4187–4193, 2007.
- [9] G. H. Golub and C. F. V. Loan, *Matrix Computations*, The Johns Hopkins University Press, 3rd edition, 1996.
- [10] I. Jolliffe, *Principal Component Analysis*, Springer, 2nd edition, 2002.
- [11] J. E. Fowler, "Compressive-projection principal component analysis," *IEEE Transactions on Image Processing*, vol. 18, no. 10, pp. 2230–2242, 2009.
- [12] P. Drineas and M. W. Mahoney, "A randomized algorithm for a tensor-based generalization of the singular value decomposition," *Linear Algebra and Its Applications*, vol. 420, no. 2-3, pp. 553–571, 2007.
- [13] L. Trefethen and D. Bau, *Numerical Linear Algebra*, Society For Industrial Mathematics, 1997.
- [14] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.
- [15] Q. Zhang, R. Plemmons, D. Kittle, D. Brady, and S. Prasad, "Joint segmentation and reconstruction of hyperspectral data with compressed measurements," *Applied Optics*, vol. 50, no. 22, pp. 4417–4435, 2011.
- [16] M. E. Gehm, R. John, D. J. Brady, R. M. Willett, and T. J. Schulz, "Single-shot compressive spectral imaging with a dual-disperser architecture," *Optics Express*, vol. 15, no. 21, pp. 14013–14027, 2007.

- [17] A. Wagadarikar, R. John, R. Willett, and D. Brady, "Single disperser design for coded aperture snapshot spectral imaging," *Applied Optics*, vol. 47, no. 10, pp. B44–B51, 2008.
- [18] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Effects of linear projections on the performance of target detection and classification in hyperspectral imagery," *Journal of Applied Remote Sensing*, vol. 5, no. 1, Article ID 053563, 2011.
- [19] G. Martinsson, "Randomized methods for computing the singular value decomposition of very large matrices," in *Workshop on Algorithms for Modern Massive Data Sets*, 2012.
- [20] A. D. Meigs, L. J. Otten, and T. Y. Cherezova, "Ultraspectral imaging: a new contribution to global virtual presence," in *Proceedings of the IEEE Aerospace Conference*, vol. 2, pp. 5–12, March 1998.
- [21] A. Berk, G. P. Anderson, P. K. Acharya et al., "MODTRAN 5, a reformulated atmospheric band model with auxiliary species and practical multiple scattering options: update," in *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XI*, vol. 5806 of *Proceedings of SPIE*, pp. 662–667, 2005.
- [22] B. Parlett, *The Symmetric Eigenvalue Problem*, vol. 20, Society for Industrial Mathematics, 1998.
- [23] M. A. O'Neil and M. Burtscher, "Floating-point data compression at 75 Gb/s on a GPU," in *Proceedings of the 4th Workshop on General Purpose Processing on Graphics Processing Units (GPGPU '11)*, p. 7, March 2011.
- [24] R. C. Olsen, S. Bergman, and R. G. Resmini, "Target detection in a forest environment using spectral imagery," in *Imaging Spectrometry III*, vol. 3118 of *Proceedings of SPIE*, pp. 46–56, July 1997.
- [25] C. I. Chang, "Target signature-constrained mixed pixel classification for hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 5, pp. 1065–1081, 2002.
- [26] B. Thai and G. Healey, "Invariant subpixel material detection in hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 3, pp. 599–608, 2002.



Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

